ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

INTEGRACION DE REDES FERROVIARIAS EN BUSCADOR MULTI-TRANSPORTE ENFOCADO EN LA REGION DEL MEKONG

Autor: David Luengo Juárez

Director: Enrique Perez Martin

Madrid

Junio 2018

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Integración de redes ferroviarias en buscador multi-transporte enfocado en la región del Mekong

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico **2017/18** es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: David Luengo Juárez Fecha: 25/01/2019

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Enrique Pérez Martin Fecha: 25/01/2019

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. **David Luengo Juárez**

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: **Integración de redes ferroviarias en buscador multi-transporte enfocado en la región del Mekong**, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente)*.

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 25 de Enero de 2019

ACEPTA

Fdo David Luengo Juárez

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

INTEGRACION DE REDES FERROVIARIAS EN BUSCADOR MULTI-TRANSPORTE ENFOCADO EN LA REGION DEL MEKONG

Autor: David Luengo Juárez

Director: Enrique Pérez Martin

Madrid

Junio 2018

Agradecimientos

Quiero agradecer a Baolau el haberme dado la oportunidad de trabajar con ellos para desarrollar este proyecto.

También me gustaría agradecerles a mis padres todo el esfuerzo y sacrificio que han hecho durante todos estos años y por haberme apoyado en todo momento para que llegue donde estoy ahora, esto va por vosotros.

Por último, también agradecer a toda mi familia el apoyo que he recibido durante estos años.

Muchas gracias.

INTEGRACION DE REDES FERROVIARIAS EN BUSCADOR MULTI-TRANSPORTE ENFOCADO EN LA REGION DEL MEKONG

Autor: Luengo Juárez, David.
Director: Pérez Martin, Enrique.
Entidad Colaboradora: Baolau

RESUMEN DEL PROYECTO

Este proyecto se centra en el desarrollo de un sistema software para integrar nodos y estaciones de una red ferroviaria en un buscador multi-transporte. El sistema desarrollado permite automatizar la extracción de datos de dichos nodos y estaciones asi como la gestión de la compra del billete por parte del usuario.

Palabras clave: Automatizacion, *web scrapping*, multi-buscador

1. Introducción

Con el auge del turismo en el sudeste asiático y el creciente uso de los trenes como medio de transporte económico, Baolau vio la oportunidad de incluir la red de ferrocarril de Malasia a su buscador para que asi los viajeros pudiesen reservar sus billetes online garantizando la reserva antes de viajar y sin tener que ir a la estación.

En la actualidad, la reserva de billetes de forma online es proporcionada por KTM, principal operador de la red ferroviaria del país. Con el desarrollo y la consecuente integración del sistema software descrito en este documento, los viajeros tendrán la posibilidad de planificar su viaje en tren por Malasia. Con este proyecto, tambien se brinda la oportunidad de expandir Baolau por el resto de la región del Mekong y asi aumentar el número de clientes y reservas.

2. Definición del Proyecto

Este proyecto consiste en el desarrollo de un sistema software *back-end* para realizar una serie de tareas autónomas con poca o nula interacción por parte del personal humano. Por una parte, el sistema extraerá la información necesaria sobre una estación de tren de Malasia: coordenadas, idiomas… y, por otra parte, el sistema será capaz de integrarse de una forma eficiente y sencilla al proceso de reserva y compra del billete del buscador.

3. Descripción del sistema

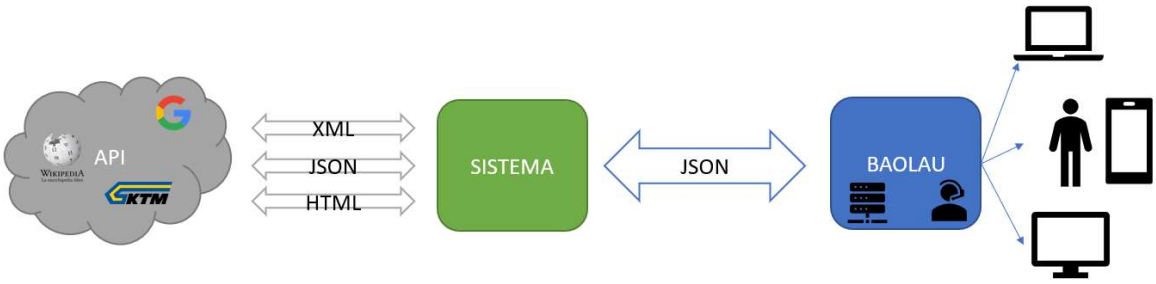


Ilustración 1. Descripcion del sistema

El sistema software propuesto se puede entender como dos subsistemas diferentes.

El primer subsistema se encarga de obtener, organizar y crear la información necesaria sobre las estaciones de la red de ferrocarril de una forma automática. Antes de hacer uso de este primer subsistema, se hace una lista con los nombres de las estaciones o las ciudades que se quieren incluir al buscador. Una vez obtenido esto, se introduce al sistema para buscar los idiomas y coordenadas de dichas estaciones. Mas adelante, se organiza esta información de tal forma que quede en el mismo formato que en la base de datos del buscador. Al finalizar este proceso, se crean las rutas operacionales entre las distintas estaciones. Las rutas operacionales sirven para crear el vinculo entre Baolau y el proveedor de la venta de billetes.

El segundo subsistema se ocupa de hacer el proceso de reserva. Este proceso se divide en cuatro partes diferentes. La primera de ellas se encarga de buscar la información de los trenes entre el origen y destino (horarios, precios, disponibilidad...). La segunda parte obtiene la información necesaria para construir el mapa de asientos. Las dos ultimas partes se encargan de reservar el asiento y emitir el billete respectivamente.

4. Resultados

Los resultados obtenidos tras el desarrollo del sistema son satisfactorios. En las siguientes figuras se muestran algunos ejemplos tras ejecutarse el software.


```
{
  "latitude": "4.59906255",
  "longitude": "101.073891452714",
  "urlSearch": "https://nominatim.openstreetmap.org/search?q=Ipoh&countrycodes=MY&accept-language=en-US&format=json&addressdetails=1",
  "message": "OK"
}
```


Figura 1. Resultados extraccion de coordenadas

```
{
  "English": "Ipoh",
  "Japanese": "イポー",
  "Chinese": "怡保",
  "Simplified Chinese": "怡保",
  "Traditional Chinese": "怡保"
}
```


Figura 2. Resultados extraccion de idiomas


```
{
  "origin_id": 1518,
  "destination_id": 1519,
  "origin_code": "180:2",
  "destination_code": "181:2",
  "transport_code": "ebk",
  "deleted": 0
},
{
  "origin_id": 1518,
  "destination_id": 1526,
  "origin_code": "180:2",
  "destination_code": "300:2",
  "transport_code": "ebk",
  "deleted": 0
},
```


Figura 4. Resultados creacion de ruta operacional


```
{
  "id": 1515,
  "node_type": "train",
  "name_en": "Ipoh Railway Station",
  "name_vi": "Ga Ipoh",
  "name_ja": "イポー駅",
  "name_cn": "怡保站",
  "name_tw": "怡保站",
  "town_en": "IPOH",
  "town_vi": "IPOH",
  "town_ja": "イポー",
  "town_cn": "怡保",
  "town_tw": "怡保",
  "country_code": "MY",
  "latitude": "4.59906255",
  "longitude": "101.073891452714",
  "deleted": 0
},
```


Figura 3. Resultados creacion de nodo

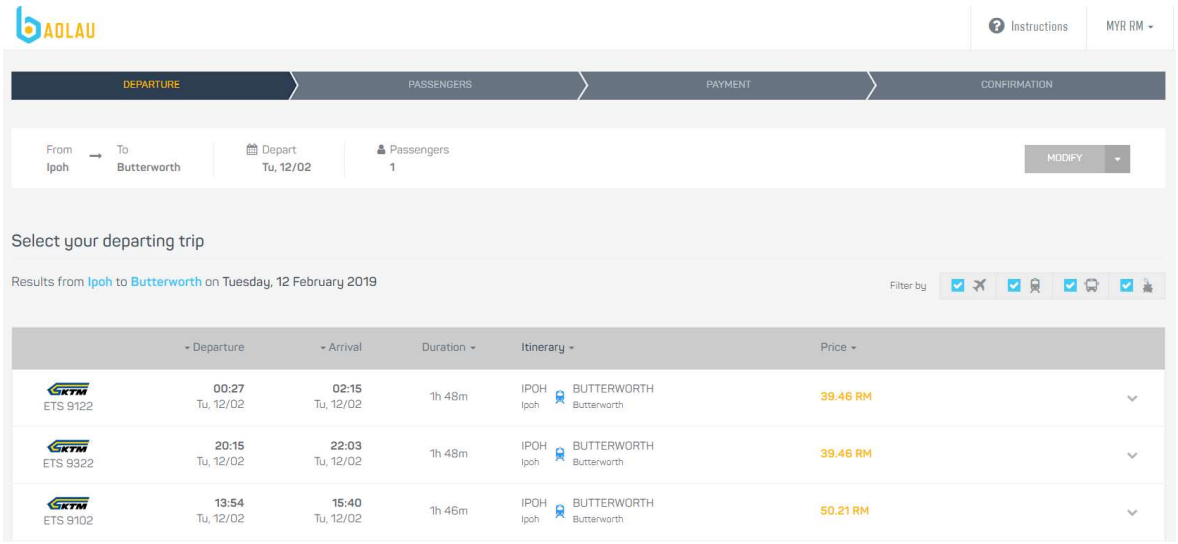


Figura 5. Resultados de las rutas

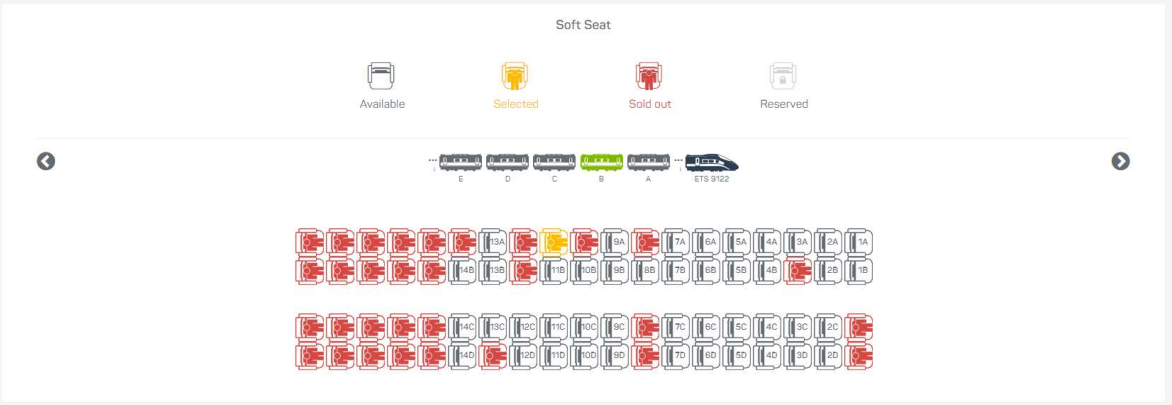


Figura 6. Resultados mapa de asientos

5. Conclusiones

Tras la implementacion y verificación del sistema mediante pruebas, se concluye que el sistema funciona atendiendo a los requisitos y objetivos propuestos.

6. Referencias

[1] Media Wiki API help. https://www.wikidata.org/w/api.php.

[2] Web Scraping. https://es.wikipedia.org/wiki/Web_scraping.

[3] CodeIgniter Active Record. https://codeigniter.com/userguide2/database/active_record.html

[4] cURL. https://curl.haxx.se

[5] Railway travel blog. https://railtravelstation.com/

[6] HTML to DOM. http://simplehtmldom.sourceforge.net/manual.htm

[7] Learning PHP. https://laracasts.com/series/php-for-beginners

INTEGRATION OF RAILWAY NETWORKS IN MULTI-TRANSPORTATION SEARCH ENGINE ENGAGED IN THE MEKONG REGION

Author: Luengo Juárez, David.
Supervisor: Pérez Martin, Enrique.
Collaborating Entity: Baolau.

ABSTRACT

This Project focuses on the development of a software system that integrates nodes and stations of a railway network in a multi-transport search engine. The developed system allows to automatize the extraction of data from said nodes and stations as well as the management of the booking system.

Keywords: Automatization, web scrapping, multi-search engine

1. Introduction

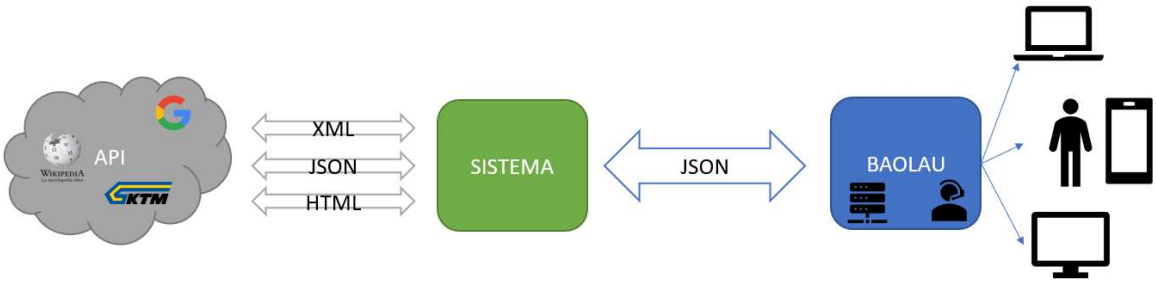
With the rise of tourism in Southeast Asia and the increasing use of trains as a means of economic transport, Baolau saw the opportunity to include the Malaysian railway network to its search engine so that travelers could book their tickets online guaranteeing the reservation before traveling and without having to go to the station.

At present, online ticket reservation is provided by KTM, the main operator of the country's rail network. With the development and the consequent integration of the software system described in this document, travelers will have the possibility to plan their trip by train through Malaysia. With this project, there is also the opportunity to expand Baolau in the rest of the Mekong region and thus increase the number of clients and reservations.

2. Project definition

This project consists in the development of a back-end software system to perform a series of autonomous tasks with little or no interaction of human personnel. On the one hand, the system will extract the necessary information about a train station in Malaysia: coordinates, languages ... and, on the other hand, the system will be able to integrate in an efficient and simple way the process of booking and buying the ticket of the search engine.

3. System description



Ilustration 2. System description

The proposed software system can be understood as two different subsystems.

The first subsystem is responsible for obtaining, organizing and creating the necessary information about the stations of the railway network in an automatic way. Before making use of this first subsystem, a list is made with the names of the stations or cities that are to be included in the search engine. Once this has been obtained, it is introduced to the system to search for the languages and coordinates of said stations. Later, this information is organized in such way that it remains in the same format as in the search engine's database. At the end of this process, the operational routes between the different stations are created. The operational routes serve to create the link between Baolau and the supplier of ticket sales.

The second subsystem is responsible for making the reservation process. This process is divided into four different parts. The first one oversees searching the information of the trains between the origin and destination (schedules, prices, availability ...). The second part gets the necessary information to build the seat map. The last two parties are responsible for reserving the seat and issuing the ticket respectively.

4. Results

The results obtained after the development of the system are satisfactory. The following figures show some examples after running the software.


```
{
  "latitude": "4.59906255",
  "longitude": "101.073891452714",
  "urlSearch": "https://nominatim.openstreetmap.org/search?q=Ipoh&countrycodes=MY&accept-language=en-US&format=json&addressdetails=1",
  "message": "OK"
}
```


Figura 7. Results of coordinates extraction

```
{
  "English": "Ipoh",
  "Japanese": "イポー",
  "Chinese": "怡保",
  "Simplified Chinese": "怡保",
  "Traditional Chinese": "怡保"
}
```


Figura 8. Results of language extraction


```
{
  "origin_id": 1518,
  "destination_id": 1519,
  "origin_code": "180:2",
  "destination_code": "181:2",
  "transport_code": "ebk",
  "deleted": 0
},
{
  "origin_id": 1518,
  "destination_id": 1526,
  "origin_code": "180:2",
  "destination_code": "300:2",
  "transport_code": "ebk",
  "deleted": 0
},
```


Figura 10. Results of operational route creation


```
{
  "id": 1515,
  "node_type": "train",
  "name_en": "Ipoh Railway Station",
  "name_vi": "Ga Ipoh",
  "name_ja": "イポー駅",
  "name_cn": "怡保站",
  "name_tw": "怡保站",
  "town_en": "IPOH",
  "town_vi": "IPOH",
  "town_ja": "イポー",
  "town_cn": "怡保",
  "town_tw": "怡保",
  "country_code": "MY",
  "latitude": "4.59906255",
  "longitude": "101.073891452714",
  "deleted": 0
},
```


Figura 9. Results of node creation

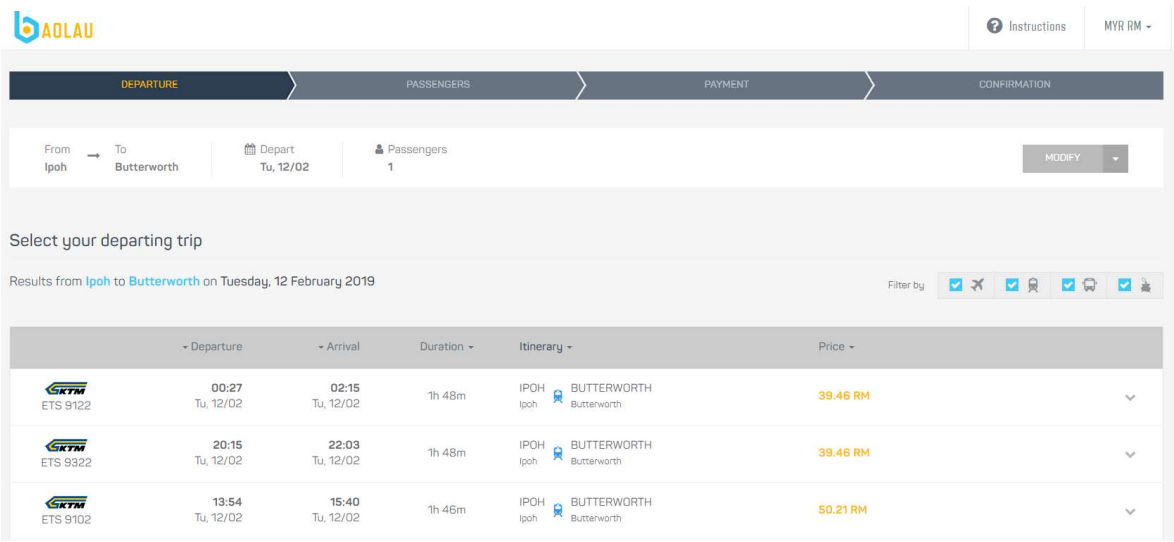


Figura 11. Resultados de las rutas

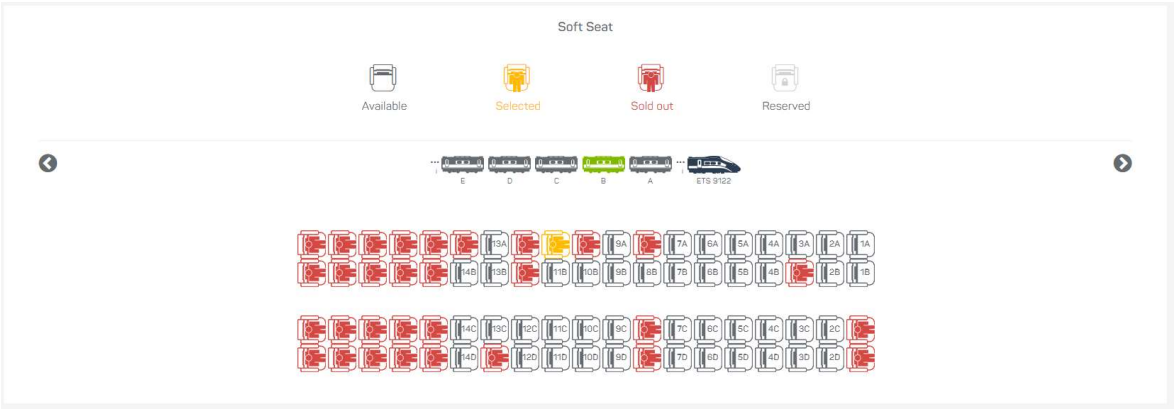


Figura 12. Resultados mapa de asientos

5. Conclusions

After the implementation and verification of the system through tests, it is concluded that the system works according to the requirements and objectives proposed.

6. References

[8] Media Wiki API help. https://www.wikidata.org/w/api.php.

[9] Web Scraping. https://es.wikipedia.org/wiki/Web_scraping.

[10] CodeIgniter Active Record. https://codeigniter.com/userguide2/database/active_record.html

[11] cURL. https://curl.haxx.se

[12] Railway travel blog. https://railtravelstation.com/

[13] HTML to DOM. http://simplehtmldom.sourceforge.net/manual.htm

[14] Learning PHP. https://laracasts.com/series/php-for-beginners

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

Este proyecto se ha desarrollado en la empresa Baolau que da un servicio de búsqueda multi-transporte.

Baolau nació con la misión de interconectar ciudades calculando tiempos y distancias, comparando rutas y facilitando la reserva de billetes para cualquier medio de transporte ya sea por tierra mar o aire. Ofrecen una plataforma que ayuda a los viajeros a encontrar su camino en la región del Mekong (Vietnam, Camboya, Laos y Tailandia) comparando las opciones de transporte y reservando los billetes cómodamente y en pocos minutos. Es una empresa que coopera con los principales proveedores de transporte de la región para integrar información precisa de horarios y precios y así poder conseguir que la reserva por internet sea accesible para todo el mundo.

En la actualidad, Baolau ofrece rutas en Tailandia, Camboya, Laos y Vietnam. Con la integración de trenes en Malasia, permitirá a los viajeros usar Baolau para encontrar los horarios y tarifas de los principales destinos en Malaysia. Se ofrecerá la reserva de billetes en más de 60 estaciones de tren distribuidas a lo largo de Malaysia incluyendo Kuala Lumpur, Johor Bahru, Gemas, Tampin, Ipoh, Butterworth…

La integración de las estaciones de tren de Malaysia permitirá conectar con los servicios de ferrocarriles del sudeste asiático, varios países como son Singapur, Tailandia, etc. a un precio bastante competitivo si se compara con otros medios de transporte como es el avión.

En este capítulo se hace una introducción a este proyecto, describiendo la motivación de este. Se hará una breve explicación sobre lo que es el *web scrapping* y sobre la importancia de este proyecto.

1.1 MOTIVACIÓN DEL PROYECTO

Este proyecto surge como necesidad para la empresa para poder seguir con su misión de interconectar ciudades en la región del Mekong calculando tiempos y distancias, comparando rutas y facilitando la reserva de billetes para cualquier medio de transporte: avión, bus, ferry y tren.

Con este proyecto, se ofrecerá mayor cobertura y una mayor oferta a los usuarios que utilizan el buscador para comprar sus billetes. También situará a Baolau como uno de los principales vendedores de billetes de tren online en Malaysia. El proyecto que se va a desarrollar también incrementará el número de reservas que se hacen utilizando el buscador y por tanto generar mas beneficios a la empresa.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

2.1 PHP

Lenguaje de programación de propósito general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinamico. El código es interpretado por un servidor web con un modulo de procesador de PHP que genera el HTML resultante.

2.2 PHPSTORM

Se trata de un IDE multiplataforma para PHP. Proporciona un editor para PHP, HTML y JavaScript con análisis de código a medida que se escribe, prevención de errores y refactorización automática para código PHP.

2.3 CODEIGNITER

Es un *framework* para aplicaciones web de código abierto para crear sitios web dinámicos con PHP. Su objetio es permitir que los desarrolladores puedan realizar proyectos mucho mas rápido que creando toda la estructura desde cero, brindando un conjunto de bibliotecas para tareas comunes, asi como una interfaz simple y una estructura lógica para acceder a esas bibliotecas.

2.4 XAMPP

Paquete de software libre que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web apache y los interpretes para lenguajes de script PHP y Perl. Actúa como un servidor web libre, fácil de usar y capaz de interpretar paginas dinámicas.

2.5 BITBUCKET

Bitbucket es un servicio de alojamiento basado en web, para proyectos que utilizan el sistema de control de versiones mercurial y git.

2.6 GIT

Es un software de control de versiones pensando en la eficacia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran numero de archivos de código fuente. Su propisito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

2.7 KATALON STUDIO

Katalon Studio es una herramienta para realizar *testings* automáticos. Ofrece una interfaz dual intercambiable para scripting: un editor tabular de grabación para los usuarios menos técnicos y un IDE de creación de scripts orientado a probadores experimentados para realizar pruebas de automatización con resaltado de sintaxis y finalización inteligente de código.

La planificación de las pruebas se puede estructurar utilizando un conjunto de pruebas con variables de entorno. La ejecución de la prueba se puede parametrizar y paralelizar utilizando perfiles.

2.8 cURL

cURL es un proyecto software orientado a la transferencia de archivos. Es una herramienta de línea de comandos para obtener o enviar archivos mediante la sintaxis URL. El principal propósito y uso para cURL es automatizar transferencias de archivos o secuencias de operaciones no supervisadas. Es una herramienta valida para simular las acciones de usuarios en un navegador web.

Capítulo 3. ESTADO DE LA CUESTIÓN

3.1 INTRODUCCIÓN

Malayan Railway (Keratapi Tanah Melayu Berhad o KTM) gestiona una red de ferrocarriles de casi 2.092 km de extensión. Existen tres tipos de trenes: primera segunda y tercera clase. El tren más rápido Express Rakyat recorre entre Singapur y Butterworth (Malaysia) y continua hasta Tailandia. Los trenes son modernos y algunos tienen literas. En el este de Malaysia existe una línea que se conoce como el Ferrocarril de la Jungla que es visitada por muchos viajeros y que constituye la principal ruta terrestre hasta el parque nacional de Taman Negara. Además de esta línea, hay otras dos. Una de ellas recorre la costa oeste desde Singapur que va hasta Kuala Lumpur y se une con la red de ferrocarril tailandesa en la frontera. La otra línea se separa de la costa occidental y se dirige al noroeste.

Gracias al programa de modernización que está llevando a cabo el gobierno de Malaysia a su red ferroviaria, se están incorporando servicios *intercity* frecuentes (16 trenes/dia). Esto está facilitando el transporte por tren en la península de Malaysia y por tanto también está aumentando el número de viajeros que utilizan este tipo de transporte.

Con el auge del turismo en el sudeste asiático y el creciente uso de los trenes, Baolau vio la oportunidad de incluir estas redes de ferrocarriles en Malaysia para que así los viajeros puedan reservar sus billetes online garantizando su reserva sin tener que ir a la estación a reservar.

En la actualidad, la reserva de billetes de forma online es proporcionada por KTM (principal operador de la red ferroviaria de Malaysia). Con la integración de este proyecto a la empresa, todos los clientes y viajeros que actualmente reservan sus billetes con Baolau, también podrán hacerlo para planificar su viaje en tren por Malaysia. Con esta integración, también se brinda la oportunidad de expandir Baolau por el resto de la región del Mekong y así aumentar el número de reservas de la empresa.

3.2 SOLUCIÓN

La solución es crear un subsistema que permita integrarse fácilmente al sistema ya creado sin afectarlo. Este subsistema tiene que ser capaz de gestionar las reservas que se hagan de forma automática asi como automatizar la extracción de información sobre las rutas y los nodos de dicho pais. Esto facilitara la labor del equipo de operaciones de la empresa para no realizar las reservas de forma manual.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

La finalidad que se pretende conseguir con este proyecto es la de crear una solución que permita automatizar el proceso de compra de billetes sin intervención del operador.

4.2 OBJETIVOS

4.2.1 OBJETIVOS GENERALES

El objetivo general que se persigue con este proyecto es integrar las redes ferroviarias de Malaysia al buscador multi transporte que ofrece Baolau.

Este objetivo se construye a partir de otros relacionados con la expansión de la empresa en el sudeste asiático. Debido a la creciente demanda de viajeros en el sudeste asiático, Baolau ha experimentado un aumento en el número de reservas que se hacen en el buscador. Es por ello, que, como plan de crecimiento, han decidido expandir su oferta de rutas ferroviarias a Malaysia para si interconectar Malaysia con Singapur y Bangkok y ofrecer una mayor cobertura y oferta a todos los clientes que utilizan el buscador.

4.2.2 OBJETIVOS ESPECIFICOS DEL PROYECTO

El objetivo principal de este proyecto es el de crear un sistema que automatice todo el proceso de reserva de billetes. El sistema debe ser capaz de automatizar el proceso de extracción de información de los nodos (coordenadas, idiomas…), de la ruta (horarios, precios, trenes disponibles…), de la reserva y de la creación del billete.

4.2.3 OBJETIVOS ESPECIFICOS DE LA INTEGRACIÓN

Teniendo en cuenta el objetivo principal del proyecto, existen una serie de objetivos secundarios que también se persiguen en este proyecto, que son:

- Minimizar el coste de desarrollo: esto quiere decir que los errores que puedan surgir sean detectados y solucionados a tiempo
- Maximizar la calidad: evitar que las versiones del código nuevo no estropeen al código que ya estaba implementado.

En resumen, la integración de todo el proceso desde que el usuario hace una búsqueda entre origen y destino hasta que realiza su compra, tiene que hacerse de forma fluida y automática.

4.3 METODOLOGÍA

La metodología de trabajo que se ha llevado a cabo es una metodología de trabajo incremental. Durante un periodo de cuatro semanas aproximadamente, se crea un incremento del software que es potencialmente utilizable.

El flujo de desarrollo que se ha seguido es el siguiente:

Figura 13. Metodologia de trabajo

Tareas	Responsable	Duracion
Periodo Aprendizaje		4 Semanas
Comprension de las tecnlogias	David	1 Semana
Lenguaje PHP	David	1 Semana
Comprension estructura del buscador	David	1 Semana
Mini-Integracion estaciones Tailandia	David	1 Seamana
Inicio		4 Semanas
Determinar los objetivos	David	1 Semana
Diseño	David	2 Semans
Plan de recursos	David	1 Seamana
Desarrollo		8 Semanas
Desarrollo de base de datos	David	1 Semana
Desarrollo SW	David	4 Semanas
Pruebas	David	1 Semana
Completar desarrollo	David	2 Semanas
Lanzamiento		2 Semanas
Integracion del sistema	David	2 Semanas
Operaciones		4 Semanas
Pruebas del sistema	David	4 Semanas
Documentacion		

Figura 14. Planificacion y tareas del trabajo

Capítulo 5. SISTEMA DESARROLLADO

En este capítulo se describe el sistema que se ha desarrollado desde su diseño hasta su implementación.

5.1 ANÁLISIS DEL SISTEMA

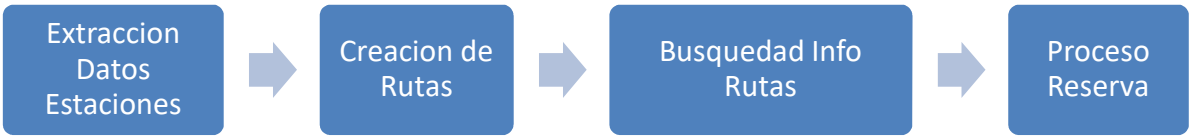


Figura 15. Analisis del sistema

5.1.1 EXTRACCIÓN DATOS ESTACIONES

El proceso de extracción de información sobre las estaciones para luego guardarlos en la base de datos del buscador se ha intentado automatizar en la medida de lo posible. Este proceso consiste en obtener información de:

- Nombres de los nodos en los distintos idiomas del buscador (chino, japonés, ingles…)
- Coordenadas de los nodos para luego mostrarlos en el mapa.

Para poder automatizar este proceso, se ha utilizado las API de Google, Opeenstreetmap y wikidata.

En algunos de los nodos, la extracción automática de información no dio resultados ya que los nodos se situaban en zonas remotas y se tuvo que buscar la información de manera manual.

Una vez extraidos los datos de las estaciones, se convierten al formato que luego se guarda en la base de datos del buscador.

5.1.2 CREACIÓN DE RUTAS

Una ruta se considera el trayecto desde un punto origen hacia un destino. Una vez obtenido los nodos/estaciones mediante el proceso explicado en el apartado anterior, se crean las conexiones entre dichos nodos. La informacion necesaria que debe tener esta ruta, tiene que ser tal que permita luego buscarla en el proveedor. Un ejemplo de los campos necesarios de la ruta son los siguientes:

- id_origen: id del nodo origen que se guarda en la base de datos de Baolau
- id_destino: id del nodo destino que se guarda en la base de datos de Baolau
- código_origen: código del nodo origen para buscar la información necesaria en el proveedor.
- código_destino: código del nodo destino para buscar la información necesaria en el proveedor.
- código_proveedor: código que identifica a cada uno de los proveedores de informacion

5.1.3 BÚSQUEDA DE INFORMACIÓN DE RUTAS

En este proceso, cuando el usuario introduce los puntos de origen y destino y las fechas en las que quiere viajar, el sistema debe obtener de manera automática toda la información necesaria sobre las rutas disponibles que hay (hora de salida, duración, numero de tren, precios por tipo de pasajeros…). Una vez obtenida esta información, se convierte al formato que se guarda en la base de datos y que luego se muestra en el buscador.

La forma en la que se busca esta información difiere entre Malaysia y Tailandia. Para el caso de Malaysia, al tener un proveedor fiable, la información se obtenia haciendo uso de su API, en la que los datos se devolvían en formato JSON.

En el caso de Thailandia, el proceso fue mas tedioso ya que se tuvo que hacer *scrapping* de la pagina web del proveedor para obtener toda la información necesaria.

El *scrapping* web se basa en obtener información de interés de una pagina web para luego convertir esa información en formatos como: JSON, XML, CSV, etc… Basicamente, se extrae la información de los sitios web simulando la navegación de un humano.

Esta busquedad de información tanto en el caso de Thailandia como en el caso de Malaysia, se hace en la función scrap descrita en el apartado 5.2 Diseño.

5.1.4 PROCESO DE RESERVA

Esta parte del sistema consiste primero en hacer la reserva, que se realiza en la función prebook descrita en el apartado 5.2 Diseño, y segundo confirmar la reserva, que se realiza en la función book.

En la función prebook, se prepara toda la información necesaria relativa a la reserva (datos de los pasajeros, información sobre los asientos seleccionados…). Una vez se preparan los datos, se hace una llamada a la API del proveedor para reservar los billetes. En esta función también se hace un manejo de los posibles errores (no se ha podido reservar los asientos, la llamada al proveedor ha fallado…).

Con la función book, se vuelve a hacer una llamada a la API del proveedor para confirmar la reserva. Una vez confirmada la reserva, se genera automáticamente el *ticket* y se guardan los datos necesarios sobre la reserva en la base de datos.

Este proceso de reserva solo se ha hecho en el sistema de Malaysia donde hubo un acuerdo con un proveedor local.

5.2 DISEÑO

En este apartado se presenta el diseño del software.

5.2.1 DISEÑO EXTRACCION DE DATOS ESTACIONES

Función	***Parámetros***	***Retornos***	***Descripción***
findNodeCoordinates	nodeName, area, countryCode, transport	array	Función que busca las coordenadas del nodo utilizando dos API (Google y Openstreetmap)
findLanguages	Name, countryCode	array	Funcion que busca los idiomas de la ciudad utilizando la API de wikidata.

Tabla 1. Funciones para la extraccion de datos

5.2.2 DISEÑO MALAYSIA

Función	*Parámetros*	*Retornos*	*Descripción*
scrap	params	array	Función que: • Busca las rutas en el formato del proveedor según el origen y destino introducido en el buscador • Obtiene las rutas del proveedor y las convierte en el formato de Baolau para guardarlas en la base de datos.
getSubplacesEBK	countryCode, newInsert	array	Función que obtiene todos los nodos origen y destino del proveedor y los convierte al formato que se guarda en la base de datos.

trainSeatMap	coachSeatPlan, coachType, ticketFare, decks	array	Función que dibuja el mapa de asientos obteniendo los datos del proveedor y simplificándolos en el formato que usa el buscador para mostrarlos.
triptoRoute	trips, ids, params	array	Función que hace la conversión de la ruta proporcionada por el proveedor en el formato de Baolau.
preBook	Booking, step	array	Función que prepara todos los datos necesarios para hacer la reserva
book	Booking, step	array	Función que realiza la reserva y genera los tickets.
generateVoucherEBK	url, results, booking, step	-	Función que genera el ticket con todos los datos necesarios de los pasajeros.
provider	Operation, data, post	array	Función que realiza todas las llamadas necesarias al proveedor según el tipo de operación que se quiera hacer.

buildDataFromEBK	countryCode, lastId, operation, params	-	Función que construye los nodos, las ciudades y las rutas operacionales para luego guardarlo en la BBDD

Tabla 2. Funciones del diseño de Malasia

5.2.3 DISEÑO TAILANDIA

Función	***Parámetros***	***Retornos***	***Descripción***
scrap	params	array	Función que: • Obtiene los horarios y los trenes desde el origen y el destino haciendo web scrapping en la pagina del proveedor • Obtiene todos los precios por cada tren • Ordena y simplifica todos los datos para guardarlos en la BBDD.

Tabla 3. Funciones diseño de Tailandia

5.3 IMPLEMENTACIÓN

En esta parte del proyecto se codifican todos los scripts descritos en el diseño. Para facilitar el entendimiento del sistema desarrollado, en este apartado se mostrará el código fuente o parte del código de los scripts más destacables del proyecto con mayor funcionalidad.

5.3.1 CÓDIGO EXTRACCIÓN DE DATOS DE ESTACIONES

```
public function findNodeCoordinates($nodeName, $area, $countryCode, $transport,
$useGoogle = true, $googleAPIKey = "")
    {
        $type2Google = array(
            "train" => "+railway+station",
            "plane" => "+airport",
            "bus"   => "+bus+station"
        );
        $type2Backup = array(
            "train" => "railway",
            "plane" => "aeroway",
            "bus"   => "bus_station",
        );

        $output = array();
        $htmlResults = array();

        $name = implode(' ', array_map('ucfirst', array_map('strtolower',
explode(' ', urldecode($nodeName)))));
        //$name = str_replace(" ", "+", strtolower(trim($name)));
        //$location = ",+" . urlencode(urldecode($area));
        //$country = ",+" .
urlencode(get_list_countries()[strtoupper($countryCode)]);
        $backup = false;

        if ($useGoogle && $googleAPIKey) {
            $maxRetries = 3;
            $retries = 0;
            $googleKey = "&key=" . $googleAPIKey;
            $searchType = (isset($type2Google[$transport]) && !stripos($nodeName,
trim(str_replace('+', " ", $type2Google[$transport]))) !== false) ?
$type2Google[$transport] : "";

            do {
                $retries++;
                if ($transport == "train") $type = "train_station";
                else if ($transport == "plane") $type = "airport";
                else if ($transport == "bus") $type = "bus_station";

                //GOOGLE API
```

```
                $urlGoogle =
"https://maps.google.com/maps/api/geocode/json?region=$countryCode&address=" .
urlencode($name . $searchType) . "&result_type=$type" . $googleKey;
                //$urlGoogle =
"https://maps.google.com/maps/api/geocode/json?region=$countryCode&address=" .
urlencode($name . $searchType . $location) . $googleKey;
                $htmlJson = json_decode($this->getHTML($urlGoogle), true);
                $htmlResults['Google'] = $htmlJson;

                if ($htmlJson['status'] == 'OK') {
                    $lastElement = end($htmlJson['results']);

                    foreach ($htmlJson['results'] as $resultCoord) {
                        //stripos( str_replace(' ', '',
$resultCoord['formatted_address']), str_replace(' ','',urldecode($nodeName))) !==
false
                        if (isset($type) && in_array($type,
$resultCoord['types']) &&
                            (stripos($resultCoord['formatted_address'],
urldecode($nodeName)) !== false
                                || stripos($resultCoord['formatted_address'],
urldecode($area)) !== false)) {
                            $buildNode['latitude'] =
isset($resultCoord['geometry']['location']['lat']) ?

(string)$resultCoord['geometry']['location']['lat'] : "";
                            $buildNode['longitude'] =
isset($resultCoord['geometry']['location']['lng']) ?

(string)$resultCoord['geometry']['location']['lng'] : "";
                            $buildNode['urlSearch'] = $urlGoogle;
                            $buildNode['message'] = "OK";
                            $output = $buildNode;
                            break 2;
                        }
                        if ((stripos($resultCoord['formatted_address'],
urldecode($nodeName)) !== false
                            || stripos($resultCoord['formatted_address'],
urldecode($area)) !== false)) {

                            $buildNode['latitude'] =
isset($resultCoord['geometry']['location']['lat']) ?

(string)$resultCoord['geometry']['location']['lat'] : "";
                            $buildNode['longitude'] =
isset($resultCoord['geometry']['location']['lng']) ?

(string)$resultCoord['geometry']['location']['lng'] : "";
                            $buildNode['urlSearch'] = $urlGoogle;
                            $buildNode['message'] = "CHECK COORDINATES";
                            $output = $buildNode;
                            break 2;
                        }
```

```
                        if ($resultCoord == $lastElement) {
                            $backup = true;
                            break 2;
                        }
                    }
                } /*else if ($htmlJson['status'] == 'OVER_QUERY_LIMIT') {
                    sleep(2);
                    $googleKey = ($retries == ($maxRetries - 1)) ? "&key=" .
$googleAPIKey : "";
                    //continue;
                }*/ else {
                    log_message("error", "[info] Google not successful: " .
$htmlJson['status'] . " trying backup...");
                    $backup = true;
                    break;
                }
            } while ($retries <= $maxRetries);
        }

        if (!$useGoogle || $backup || empty($googleAPIKey)) {
            $urlBackUp =
"https://nominatim.openstreetmap.org/search?q=$name&countrycodes=$countryCode&acc
ept-language=en-US&format=json&addressdetails=1";
            $backup = false;
            $this->use_proxy = false;
            $html = $this->getHTML($urlBackUp);
            if (!$htmlJson = json_decode($html, true)) $htmlJson = array();
            $htmlResults['Backup'] = $htmlJson;
            $lastElement = end($htmlJson);
            foreach ($htmlJson as $place) {
                if ($place['class'] == $type2Backup[$transport] /* &&
$place['type'] == "station" */) {
                    $buildNode['latitude'] = (string)$place['lat'];
                    $buildNode['longitude'] = (string)$place['lon'];
                    $buildNode['urlSearch'] = $urlBackUp;
                    $buildNode['message'] = "OK";
                    $output = $buildNode;
                    break;
                }
                if ($place == $lastElement) {
                    $buildNode['latitude'] = "";
                    $buildNode['longitude'] = "";
                    $buildNode['urlSearch'] = json_encode($htmlResults);
                    $buildNode['message'] = "NOT FOUND";
                    $output = $buildNode;
                }
            }
        }
        if (empty($output)) {
            $buildNode['latitude'] = "";
            $buildNode['longitude'] = "";
            $buildNode['urlSearch'] = json_encode($htmlResults);
            $buildNode['message'] = "NOT FOUND";
```

```
            $output = $buildNode;
        }

        return $output;
    }
    public function findLanguages($name, $countryCode)
    {
        $searchLanguages = array(
            'en'      => "English",
            'ja'      => "Japanese",
            'zh'      => "Chinese",
            'zh-hans' => "Simplified Chinese",
            'zh-hant' => "Traditional Chinese",
        );

        if (!$name) {
            foreach ($searchLanguages as $langCode => $language)
                $empty[$language] = " ";
            return $empty;
        }

        //$paramLanguages = implode('|',array_keys($searchLanguages));
        $search = urlencode(implode(' ', array_map('ucfirst',
array_map('strtolower', explode(' ', urldecode($name))))));
        $urlAPI = "https://www.wikidata.org/w/api.php?";
        $urlSearchId = $urlAPI . "action=wbsearchentities&format=json&search=" .
$search . "&language=en";

        $htmlId = json_decode($this->getHTML($urlSearchId), true);

        $id = "";
        $noId = false;
        if (intval($htmlId['success']) && isset($htmlId['search'])) {
            foreach ($htmlId['search'] as $searchresult) {
                if (isset($searchresult['label']) /*&&
stripos($searchresult['label'], "railway station") !== false*/) {
                    $id = $searchresult['id'];
                    $noId = false;
                    break;
                } elseif (isset($searchresult['description']) &&
strpos($searchresult['description'],
get_list_countries()[strtoupper($countryCode)]) !== false) {
                    $id = $searchresult['id'];
                    $noId = false;
                    break;
                } else
                    $noId = true;
            }
        }

        $urlWiki = $noId ? $urlAPI .
"action=wbgetentities&sites=enwiki&titles=$search&format=json" : $urlAPI .
"action=wbgetentities&format=json&ids=$id";
```

```
        $html = json_decode($this->getHTML($urlWiki), true);
        $htmls[$urlWiki] = $html;

        if (!isset($html) || !isset($html['entities']) ||
isset($html['entities']['-1'])) {
            foreach ($searchLanguages as $langCode => $language) {
                if ($langCode == 'en')
                    $notFound[$language] = urldecode($name);
                else
                    $notFound[$language] = " ";
            }
            return $notFound;
        }

        $key = array_keys($html['entities'])[0];
        $languages = $html['entities'][$key]['labels'];

        foreach ($searchLanguages as $langCode => $language) {
            $result[$language] = isset($languages[$langCode]) ?
$languages[$langCode]['value'] : "";
        }

        if (empty($result[$searchLanguages['zh-hans']]) &&
empty($result[$searchLanguages['zh-hant']]) &&
!empty($result[$searchLanguages['zh']])) {
            $result[$searchLanguages['zh-hans']] = $result[$searchLanguages['zh-
hant']] = $result[$searchLanguages['zh']];
        }
        if (empty($result[$searchLanguages['en']]))
$result[$searchLanguages['en']] = $name;

        return $result;
    }
```

5.3.2 CÓDIGO MALAYSIA

```
<?php

defined('BASEPATH') or exit('No direct script access allowed');

trait ScrapLibEBK {

    private $typeOfTransport = 'train';

    private $currency = "SGD";

    private $type2Fare = array(
        "BERTH"     => "AnT1#AnT2",
        "SEAT"      => "NML",
        "B"         => "AnT1#AnT2",
        "S"         => "NML",
        "C"         => "Cabin",
        "WINDOW"    => "NML",
        "AISLE"     => "NML",
        "L_BERTH"   => "AnT1",
        "U_BERTH"   => "AnT2" );

    private $fare2Type = array(
        "NMLWindow" => "WINDOW",
        "NMLAisle"  => "AISLE",
        "AnT1"      => "L_BERTH",
        "AnT2"      => "U_BERTH",
        "NML"       => "WINDOW#AISLE" );

    private $notInclude=array(/*'KLIA','KLIA 2',*/'Gurun','Batu Enam','Bertam
Baru','Bukit Abu',
        'Chegar Perah','Kemubu',/*'Kerambit',*/'Limau Kasturi','Manek
Urai','Padang Tungku','Temangan' );

    /*
     * Test Data BUS
     *
     * Johor,MY (7) -> Penang, MY (5)
     *
     * Cameron Highlands,MY (12) -> Kuala Lumpur,MY (2)
     *
     */

    /*
     * Test Data TRAIN
     *
     * Node: IPOH (71) -> Node: Butterworth (30)
     *
     * Town: Perak (6) -> Town: Penang (5)
     *
     */
```

```
    /**
     * Description: Function called by Lscrap to obtain the routes
     */
    public function scrapEBK($params) {

        /* Node to EBK */
        $itinerary = $this->nodesToebk($params);

        /* Search for trips in API */
        $trips = false;
        if ($itinerary)
            $trips = $this->getTrips($itinerary);

        /* Trips to Routes */
        if ($trips) {
            $routes = array();
            $ids = $this->code2id($params['origin'], $params['destination'],
'ktm');
            $routes=array();
            // Group each trip by train
            foreach ($trips as $v)
                $trains[$v['TrainCode'].$v['DepartureDate']][] = $v;
            foreach ($trains as $train)
                if( $route = $this->tripToRoute($train, $ids, $params))
                    $routes[] = $route;
            //log_message("error", 'Scrap Routes: ' . log2url($routes,
'routes'));
            return $routes;
        } else
            return array();
    }

    /**
     * Description: Function that searches on a multidimensional array
     * @param  array $needle
     * @param  array $haystack
     * @return array $keys
     */
    private function multi_array_search($needles, $haystack,$returnvalues=false)
{

        $output = array();
        foreach ($haystack as $key => $value) {
            for ($i = 0; $i < count($needles); $i ++) {
                foreach ($needles[$i] as $k => $v)
                    if (! isset($value[$k]) || $value[$k] != $v)
                        continue 2;

                        $output[] = $returnvalues?$value:$key;
            }
        }
        return $output;
```

```
    }

    /**
     * Description: Function that get the subplaces from the API and builds the
nodes
     * @param  array  $params array that has the name of the towns to build the
nodes
     * @param  string $countryCode
     * @return array $nodes
     */
    private function getSubplacesEBK($countryCode,$newInsert=false) {
        set_time_limit(180); // TODO ELIMINAR

        $output=array();

        if (! $objRoutes = $this->easyBook("routes") ) return false;

        $output['routesEBK']=$objRoutes['Routes'];

        foreach ($objRoutes['Routes'] as $route) {
            $from[]=array(
                "PlaceId"=>$route['FromPlaceId'],
                "SubPlaceId"=>$route['FromSubPlaceId'] );
            $to[]=array(
                "PlaceId"=>$route['ToPlaceId'],
                "SubPlaceId"=>$route['ToSubPlaceId'] );
        }

        $routesEBK=array_merge($from,$to);
        $routesEBK = array_values(array_unique($routesEBK, SORT_REGULAR));

        if( ! $objSubplaces = $this->easyBook("nodes") )return false;
        $keysSubplaces=$this->multi_array_search($routesEBK,
$objSubplaces['Subplaces']);


        foreach ($keysSubplaces as $key) {
            // Group each node by country
            if ($objSubplaces['Subplaces'][$key]['CountryCode'] == $countryCode)
{

                if( !
in_array(strtolower($objSubplaces['Subplaces'][$key]['SubPlaceName']),
array_map('strtolower', $this->notInclude) ) ) {
                    $nodes[] = array(
                        "town_en"      =>
$objSubplaces['Subplaces'][$key]['SubPlaceName'],
                        "town_id"      =>
$objSubplaces['Subplaces'][$key]['SubPlaceId'],
                        "state_en"     =>
$objSubplaces['Subplaces'][$key]['PlaceName'],
                        "state_id"     =>
$objSubplaces['Subplaces'][$key]['PlaceId'],
```

```
                        "latitude"     =>
$objSubplaces['Subplaces'][$key]['Latitude'],
                        "longitude"    =>
$objSubplaces['Subplaces'][$key]['Longitute'],
                        "country_code" =>
$objSubplaces['Subplaces'][$key]['CountryCode'] );
                }
                if($newInsert) {
                    $nodes[] = array(
                        "town_en"      =>
$objSubplaces['Subplaces'][$key]['SubPlaceName'],
                        "town_id"      =>
$objSubplaces['Subplaces'][$key]['SubPlaceId'],
                        "state_en"     =>
$objSubplaces['Subplaces'][$key]['PlaceName'],
                        "state_id"     =>
$objSubplaces['Subplaces'][$key]['PlaceId'],
                        "latitude"     =>
$objSubplaces['Subplaces'][$key]['Latitude'],
                        "longitude"    =>
$objSubplaces['Subplaces'][$key]['Longitute'],
                        "country_code" =>
$objSubplaces['Subplaces'][$key]['CountryCode'] );
                }
            }
        }

        $output['nodes']=$nodes;

        return $output;
    }

    /**
     * Description: Converts the parameters on the search results into Easy
Book's parameters
     * @param  array $params parameters received from search results
     * @return array $itinerary needed to search in the provider
     */
    private function nodesToebk($params) {

        list ($fromSubplace, $fromPlace) = explode(':', $params['origin']);
        list ($toSubplace, $toPlace) = explode(':', $params['destination']);
        $returnDate = date('Y-m-d', strtotime($params['departure_date'] . ' +1
day'));

        $itinerary = array(
            "FromPlaceId"       => $fromPlace,
            "ToPlaceId"         => $toPlace,
            "FromSubPlaceId"    => $fromSubplace,
            "ToSubPlaceId"      => $toSubplace,
            "DepartureDate"     => $params['departure_date'],
            "ReturnDate"        => $returnDate,
            "Currency"          => $this->currency,
```

```
            "Pax"               => $params['passengers_count']
        );
        return $itinerary;
    }

    /**
     * Description: Function that calls the API to get the trips for the given
itinerary specified in $params
     * @param  array $params itinerary in API format
     * @return array returns false if no trips are found or if there has been an
error on the API
     */
    private function getTrips($params) {
        if (! $objTrips = $this->easyBook("scrap", $params, true) ) return false;
        if ( empty($objTrips['Trips']) ) return false;
        else
            return $objTrips['Trips'];
    }

    /**
     * Description: Function that receives the seat info from the API and builds
the seat map
     * @param  string  $coachSeatPlan seat plan received from the API
     * @param  string  $coachType type of coach (S,B or C)
     * @param  array   $ticketfare
     * @param  array   $decks
     * @param  boolean $left if true, the seat map is drawn from left to right
     * @return array returns the seat info in Baolau format
     */
    private function trainSeatMap($coachSeatPlan, $coachType, $ticketfare,
$decks, $left = false) {
        $seat_info;
        $seats;
        $seat_map;
        $totalSeatColumn = 0;
        $totalSeatRow = 0;
        $floor = 0;

        $seatplan = explode('+', $coachSeatPlan);

        if ( in_array($coachType, array('S','SEAT')) ) {
            foreach ($decks as $deck) {
                $totalSeatColumn = $deck['TotalSeatColumn'] ?
$deck['TotalSeatColumn'] + 1 : intval($seatplan[0] + $seatplan[1] + 1);
                $totalSeatRow = count($deck['Seats']) / ($totalSeatColumn - 1);

                foreach ($deck['Seats'] as $seat) {
                    $row = intval($seat['SeatColumn'] - 1);
                    $column = $left ? intval($seat['SeatRow'] - 1) :
intval($totalSeatRow - $seat['SeatRow']);

                    // Update seatId when aisle is inserted
                    if ( $row == $seatplan[0] )
```

```
                        $seatColId = intval($seat['SeatColumn']);
                    else
                        intval($seat['SeatColumn']) == intval($seatplan[0] +
$seatplan[1]) ? $seatColId = intval($seat['SeatColumn']) : $seatColId =
intval($seat['SeatColumn'] - 1);

                    // Save the seat map
                    $seat_map[$floor][$row][$column] = array(
                        "type"              => "soft_seat_right",
                        "number"            => $seat['SeatLabel'],
                        //"price"             =>
intval($ticketfare[$seat['SeatType']]['AdultFare']* 100),
                        "status"            => strcmp($seat['Status'], "A") == 0
? "available" : "sold_out",
                        "seatId"            => $deck['DeckCode'] . "-$floor-" .
$seatColId . "-" . $column,
                        "booking_params"    => array(
                            "seat_number"   => $seat['SeatLabel'],
                            "seat_type"     => $seat['SeatType'],
                            "deck_code"     => $deck['DeckCode'],
                            'LoaiCho'       => $this-
>type2Fare[$seat['SeatType']],
                            'ticketPrice'   => array(
                                'ADULT'     =>
$ticketfare[$seat['SeatType']]['AdultFare'],
                                'CHILD'     =>
$ticketfare[$seat['SeatType']]['ChildFare']),
                        ) );
                }

                // Insert false seats to draw aisle on train map
                array_splice($seat_map[$floor], $seatplan[0], 0, array(
array_fill(0, $totalSeatRow, false) ) );

                $seat_info[$deck['DeckCode']] = array(
                    'type'          => "soft_seats",
                    'seat_map'      => $seat_map,
                    'coach_num'     => $deck['DeckCode'],
                    'dimensions'    => array($floor + 1, $totalSeatColumn,
$totalSeatRow )
                    );
            }
        }
        else if (in_array($coachType, array('B','BERTH')) ) {
            foreach ($decks as $deck) {
                $walls2insert =  0/*
count($deck['Seats'])/intval($seatplan[0]+$seatplan[1])-1 */ ;
                $totalSeatRow = count($deck['Seats']) / intval($seatplan[0]) +
$walls2insert;
                $contRow0 = $totalSeatRow - 1;
                $contRow1 = $totalSeatRow - 1;

                foreach ($deck['Seats'] as $seat) {
```

```
                    if ($seat['SeatColumn'] <= 2) {
                        $row = 0;

                        // Save the seat map
                        $seat_map[$floor][$row][$contRow0] = array(
                            "type"              => $seat['SeatType'] == $this-
>fare2Type['AnT1'] ? "long_seat_right" : "long_seat_left",
                            "number"            => $seat['SeatLabel'],
                            //"price"             =>
intval($ticketfare[$seat['SeatType']]['AdultFare']* 100),
                            "status"            => strcmp($seat['Status'], "A")
== 0 ? "available" : "sold_out",
                            "seatId"            => $deck['DeckCode'] . "-$floor-"
. $row . "-" . $contRow0,
                            "booking_params"    => array(
                                "seat_number"   => $seat['SeatLabel'],
                                "seat_type"     => $seat['SeatType'],
                                "deck_code"     => $deck['DeckCode'],
                                'LoaiCho'       => $this-
>type2Fare[$seat['SeatType']],
                                'ticketPrice'   => array(
                                    'ADULT'     =>
$ticketfare[$seat['SeatType']]['AdultFare'],
                                    'CHILD'     =>
$ticketfare[$seat['SeatType']]['ChildFare']),
                            )
                        );
                        $contRow0 --;
                    }
                    else {
                        $row = 1;

                        // Update seatId when aisle is inserted
                        $seatRowId = 2;

                        // Save the seat map
                        $seat_map[$floor][$row][$contRow1] = array(
                            "type"              => $seat['SeatType'] == $this-
>fare2Type['AnT1'] ? "long_seat_right" : "long_seat_left",
                            "number"            => $seat['SeatLabel'],
                            //"price"             =>
intval($ticketfare[$seat['SeatType']]['AdultFare']* 100),
                            "status"            => strcmp($seat['Status'], "A")
== 0 ? "available" : "sold_out",
                            "seatId"            => $deck['DeckCode'] . "-$floor-"
. $seatRowId . "-" . $contRow1,
                            "booking_params"    => array(
                                "seat_number"   => $seat['SeatLabel'],
                                "seat_type"     => $seat['SeatType'],
                                "deck_code"     => $deck['DeckCode'],
                                'LoaiCho'       => $this-
>type2Fare[$seat['SeatType']],
                                'ticketPrice'   => array(
```

```
                                    'ADULT'     =>
$ticketfare[$seat['SeatType']]['AdultFare'],
                                    'CHILD'     =>
$ticketfare[$seat['SeatType']]['ChildFare']),
                                )
                            );
                            $contRow1 --;
                        }
                    }

                    // Insert false seats to draw aisle on train map
                    array_splice($seat_map[$floor], 1, 0, false);

                    $seat_info[$deck['DeckCode']] = array(
                        'type'          => "soft_seats",
                        'seat_map'      => $seat_map,
                        'coach_num'     => $deck['DeckCode'],
                        'dimensions'    => array( $floor + 1, 3, $totalSeatRow )
                    );
                }
            } else if ($coachType == "C") {
                return false;
            } else
                return false;

            return $seat_info;
        }

        /**
         * Description: Function that converts the trip received from API to Baolau
route
         * @param  array $trip trip received from the API
         * @param  array $ids useful ids in order to build the route
         * @return array $route Baolau route
         */
        private function tripToRoute($trips, $ids, $params) {
            $prices = array();
            $check_availability = array();
            $faresInfo=array();
            $currency=false;

            foreach ($trips as $trip) {

                $start_timestamp = strtotime($trip['DepartureDate']);
                $start_time = date('H:i:s', $start_timestamp);
                $duration =  (!$this->ci->is_production && !$trip['TripDuration']) ?
"01:00" : $trip['TripDuration'] ; //avoid errors in test mode :P always 1 h
                //log_message('error','Duration:
'.$trip['TripDuration'].log2url($trip));

                if(strpos($trip['TrainName'], "ETS") !== false) {
                    $trainName = $trip['TrainCode']." - ".$trip['TrainName'];
```

```
                $vessel = "ETS"." ".$trip['TrainCode'] /* . " (" . date('H:i',
$start_timestamp) . ")" */;
            }
            else {
                $trainName = $trip['TrainCode']." - ".$trip['TrainName'];
                $vessel = explode(' ',$trip['TrainName'])[0] ."
".$trip['TrainCode'] /* . " (" . date('H:i', $start_timestamp) . ")" */;
            }

            //Rare Vessel names
            if(preg_match('@shuttle@i',$trip['TrainName']) &&
in_array($trip['TrainCode'],array('40','41','42','43','44','45'))) {
                $trainName = $trip['TrainCode']." - "."EKSPRES";
                $vessel = "EKSPRES" ." ".$trip['TrainCode'];
            }
            if(preg_match('@tren@i',$trip['TrainName'])) {
                $trainName = $trip['TrainCode']." - "."SHUTTLE";
                $vessel = "SHUTTLE" ." ".$trip['TrainCode'];
            }

            $coachName[$trip['CoachCode']]=$trip['CoachName'];

            //Check Availability for each coach type
            if ( in_array($trip['CoachType'], array('B','BERTH')) ||
$trip['CoachType'] == "C") {
                list ($fare1, $fare2) = explode('#', $this-
>type2Fare[$trip['CoachType']] );

                if ($trip['SeatAvailable'] >= $params['passengers_count']) {
                    if ($trip['SeatAvailable'] == 1) {
                        $check_availability[$fare1] = 1;
                        $check_availability[$fare2] = 0;
                    } else {
                        $check_availability[$fare1] =
floor($trip['SeatAvailable'] / 2);
                        $check_availability[$fare2] = ceil($trip['SeatAvailable']
/ 2);
                    }
                } else {
                    $check_availability[$fare1] = 0;
                    $check_availability[$fare2] = 0;
                }

                $prices[] = $fare1 . ":" . intval($trip['AdultPrice'] * 1.05 *
100);
                $prices[] = $fare2 . ":" . intval($trip['AdultPrice2'] * 1.05 *
100);
                $faresInfo[$fare1]=array(
                    "base_fare"=>intval($trip['AdultPrice'] * 100),
                    "child_base_fare"=>intval($trip['ChildPrice'] * 100),
                    "taxes_fees"=> intval($trip['AdultPrice'] * 0.05 * 100),
                    "child_taxes_fees"=> intval($trip['ChildPrice'] * 0.05 *
100),
```

```
                );
                $faresInfo[$fare2]=array(
                    "base_fare"=>intval($trip['AdultPrice2'] * 100),
                    "child_base_fare"=>intval($trip['ChildPrice2'] * 100),
                    "taxes_fees"=> intval($trip['AdultPrice2'] * 0.05 * 100),
                    "child_taxes_fees"=> intval($trip['ChildPrice2'] * 0.05 *
100),
                );

                $tripkeys['AnT1'] = $trip['TripKey'];
                $tripkeys['AnT2'] = $trip['TripKey'];
                $tripkeys['AnT'] = $trip['TripKey'];
                $coachtype['AnT'] = $trip['CoachType'];
            }
            else {
                $check_availability[$this->type2Fare[$trip['CoachType']]] =
$trip['SeatAvailable'];
                $prices[] = $this->type2Fare[$trip['CoachType']] . ":" .
intval($trip['AdultPrice'] * 1.05 * 100);
                $tripkeys[$this->type2Fare[$trip['CoachType']]] =
$trip['TripKey'];
                $coachtype[$this->type2Fare[$trip['CoachType']]] =
$trip['CoachType'];
                $faresInfo[$this->type2Fare[$trip['CoachType']]]=array(
                    "base_fare"=>intval($trip['AdultPrice'] * 100),
                    "child_base_fare"=>intval($trip['ChildPrice'] * 100),
                    "taxes_fees"=> intval($trip['AdultPrice'] * 0.05 * 100),
                    "child_taxes_fees"=> intval($trip['ChildPrice'] * 0.05 *
100),
                );
            }
            if(in_array($trip['Currency'],array('MYR','SGD')))
                $currency=$trip['Currency'];
        }

        $prices = implode('#', $prices);

        /* Prices Information */
        $arrayprices = explode("#", $prices);
        asort($arrayprices);
        $price = explode(':',array_values($arrayprices)[0])[1];

        /* Build the route */
        if($currency)
        {
            $route = array(
                'origin_id'             => $ids['origin_id'],
                'destination_id'        => $ids['destination_id'],
                'start_time'            => $start_time,
                'duration'              => strtotime("1970-01-01 " . $duration .
" UTC") / 60,
                'vessel'                => $vessel,
                'price'                 => intval($price),
```

```
                'prices'                => $prices,
                'currency'                  => $currency,
                'transport_code'        => 'ktm',
                'extra_data'            => array(
                    'fares_info'        => $faresInfo,
                    'booking_info'      => array(
                        'trip_keys'     => $tripkeys,
                        'coach_type'    => $coachtype,
                        'train_name'    => $trainName,
                        'coach_name'    => $coachName,
                        'BookingCode'   => "" // If not set, train.php throws
error
                    )
                )
            );

            isset($check_availability) ?
$route['extra_data']['check_availability'] = $check_availability : "";

            return $route;
        }
        else {
            log_message('error','Unsupported currency in EBK response
'.$trip['Currency']);
            return false;
        }
    }

    /**
     * Description: Function called to compute the child fare
     * @param  array $params
     * @return array
     */
    public function childfareEBK($params) {

        if ($params['age'] < 12 &&
isset($params['extra_data']['fares_info'][$params['selected_fare']])) {
            return array(
                'base_fare' =>
$params['extra_data']['fares_info'][$params['selected_fare']]['child_base_fare'],
                'taxes_fees' =>
$params['extra_data']['fares_info'][$params['selected_fare']]['child_taxes_fees']
,
            );
        }
        else {
            return array(
                'base_fare' =>  $params['adult_fare'],
                'taxes_fees' =>  $params['taxes_fees'],
            );
        }
    }
```

```
    /**
     * Description: Function that prepares the ticket data in order to send to
the API
     * @param  array $booking useful booking information of the passengers
     * @param  array $step useful information about the route
     * @return array $ticket information send to API in order to book the ticket
     */
    private function prepareDataEBK($booking, $step) {

        $passengers;

        for ($i = 0; $i < count($booking['passengers']); $i ++) {

            $passportdate =
unify_date_format_2_save($booking['passengers'][$i]['passport_expiration']);
            //$booking['passengers'][$i]['child'] ? $birthdate =
DateTime::createFromFormat('d/m/Y', $booking['passengers'][$i]['birthdate'])-
>format('Y-m-d') : $birthdate = "";
            $birthdate = ($booking['passengers'][$i]['birthdate']) ?
unify_date_format_2_save($booking['passengers'][$i]['birthdate']) : "";
            $passengertype= (!empty($birthdate) && get_age($birthdate,
$step['start_timestamp']) < 12 ) ? "CHILD" : "ADULT";

            /* Find selected seats */
            $seat_id = $step['seats'][$i]['key'];
            list ($deck, $floor, $row, $column) = explode('-', $seat_id);
            $selected_seat =
$step['extra_data']['seat_info'][$deck]['seat_map'][$floor][$row][$column];

            /* Passengers Info */
            if ($selected_seat['seatId'] == $seat_id) {
                $passengers[] = array(
                    "SeatNumber"        =>
$selected_seat['booking_params']['seat_number'],
                    "SeatType"          =>
$selected_seat['booking_params']['seat_type'],
                    "DeckCode"          => $deck,
                    "Name"              =>
$booking['passengers'][$i]['first_name'],
                    "PassportNumber"    =>
$booking['passengers'][$i]['passport'],
                    "PassportExpiryDate" => $passportdate,
                    "DateOfBirth"       => $birthdate,
                    "ContactNumber"     => $booking['phone'],
                    "Gender"            =>
(strtolower($booking['passengers'][$i]['title']) == 'mr') ? 'M' : 'F',
                    "PassengerType"     => $passengertype,
                    "Nationality"       =>
$booking['passengers'][$i]['country_code']
                );
                $departPaxList[]=array(
                    "SeatNumber"        =>
$selected_seat['booking_params']['seat_number'],
```

```
                    "SeatType"          =>
$selected_seat['booking_params']['seat_type'],
                    "DeckCode"          => $deck,
                    "Gender"            =>
(strtolower($booking['passengers'][$i]['title']) == 'mr') ? 'M' : 'F',
                    "PassengerType"     => $passengertype,
                );
            }
            else
                return false;
        }

        /* Ticket Info */
        $ticket = array(
            "TicketCollector"   => [
                "Salutation"    =>
ucfirst(strtolower($booking['passengers'][0]['title'])),
                "Name"          => $booking['name'],
                "Email"         => "trains@baolau.com",//$booking['email'],
                "ContactNumber" => $booking['phone'],
                "Nationality"   => $booking['passengers'][0]['country_code']
            ],
            "DepartTrip"        => [
                "TripKey"       =>
$step['extra_data']['booking_info']['trip_keys'][$step['selected_fare'][0]],
                "IsChooseSeat"  => true,
                "Passengers"    => $passengers
            ],

            "Currency"          => $this->currency
        );
        /* Check price breakdown */
        $priceData = array(
            "Currency"          => $this->currency,
            "DepartureTripKey"  =>
$step['extra_data']['booking_info']['trip_keys'][$step['selected_fare'][0]],
            "DepartPaxList"     => $departPaxList,
        );
        $breakdown = $this->easyBook("exactfare",$priceData,true);
        if(isset($breakdown['BookingFare']['PriceBreakDowns']) &&
count($breakdown['BookingFare']['PriceBreakDowns'])>1)
            $this->slack_message( $booking['code'].'Price breakdown for
EBK'.log2url( $breakdown, 'New results.', 0, 1 ) , 'ebk');

        return $ticket;
    }

    /**
     * Description: Function that gets the available seats and all the seats from
the API
     * @param  string $tripKey key needed to search in the API
     * @return array  $seats['available_seats'] and $seats['all_seats']
     */
```

```
    private function getSeatsEBK($tripKey) {
        $dataSeats = array(
            "TripKey"   => $tripKey,
            "Currency"  => $this->currency );

        if (! $objSeat = $this->easyBook("seats", $dataSeats, true) ) return
false;

        $seats['all_seats'] = $objSeat;

        if ($this->typeOfTransport == "train") {
            $availableSeats = array();
            foreach ($objSeat['Coach']['Decks'] as $key => $deck) {
                foreach ($deck['Seats'] as $seat)
                    if ($seat['Status'] == "A")
                        array_push($availableSeats, $seat);

                $objSeat['Coach']['Decks'][$key]['Seats'] = $availableSeats;
                $seats['available_seats'] = $objSeat;
            }
            return $seats;
        }
        else {
            foreach ($objSeat['BusSeatPlan']['Decks'] as $deck) {
                $availableSeats = array();
                foreach ($deck['Seats'] as $seat)
                    $seat['IsAvailable'] ? $availableSeats['Seats'][] = $seat :
0;
            }
            if (! empty($availableSeats) )
                return $availableSeats;
            else
                return false;
        }
    }

    /**
     * Description: Function called by Lscrap to obtain the extra data needed
     */
    public function extraDataEBK($route_id, $passengers_count) {

        /* Load routes model */
        $this->ci->load->model('routes_table_model');
        $route = $this->ci->routes_table_model->as_array()->get($route_id);

        $extra_data = json_decode($route['extra_data'], true);

        if (! isset($extra_data['seat_info']) ) {
            $output = $this->getExtraData($extra_data);
            $route['extra_data'] = $output['extra_data'];
        }
```

```
        //log_message("error", 'ExtraData Info: ' . log2url($route['extra_data'],
'extra_data EBK'));
        return $route;
    }

    /**
     * Description: function called by Passengers in order to build the seat map
     *
     * @param string $route_id
     * @param string $selected_fare
     * @return array $extra_data returns the extra data with the seat
informations
     */
    public function seatMapEBK($route_id, $selected_fare) {
        /* Load routes model */
        $this->ci->load->model('routes_table_model');
        $route = $this->ci->routes_table_model->as_array()->get($route_id);

        $extra_data = json_decode($route['extra_data'], true);

        $data = $this->getExtraData($extra_data, $selected_fare, true);

        $extra_data['seat_info']=$data['seat_info'];
        $extra_data['check_availability']=$data['check_availability'];

        $route['extra_data'] = $extra_data;

        //log_message("error", 'ExtraData Info: ' . log2url($route['extra_data'],
'seatmap EBK'));
        return $route['extra_data'];
    }

    /**
     * Description: Function that fills the extra data of the route needed for
the seat map
     * @param  array   $extra_data fills the extra_data with the information
needed
     * @param  string  $selected_fare saves the information according to the fare
     * @param  boolean $availability if true, the function will check for the
availability and return it in $output
     * @return array   $output returns the information needed
     */
    private function getExtraData($extra_data, $selected_fare, $availability =
false) {
        set_time_limit(60); // TODO ELIMINAR

        $check_availability = array();

        /* We call the API to get the seat information */
        if (! $coaches = $this-
>getSeatsEBK($extra_data['booking_info']['trip_keys'][$selected_fare]) ) return
false;
        $coach = $coaches['available_seats'];
```

```
        /* We check for the availability if needed */
        if ($availability) {
            foreach ($coach['Coach']['Decks'] as $deck)
                $count = array_count_values(array_column($deck['Seats'],
'SeatType'));

            if ( in_array($extra_data['booking_info']['coach_type'],
array('S','SEAT')) ) {
                foreach ($count as $fare => $num)
                    $totalSeats[] = intval($num);
                $check_availability[$this->type2Fare['Seat']] =
array_sum($totalSeats);
            } else
                foreach ($count as $fare => $num)
                    $check_availability[$this->type2Fare[$fare]] = intval($num);

            $extra_data['check_availability'] = $check_availability;
        }

        /* Seat Info for the Seat Map */
        $seat_info = $this-
>trainSeatMap($coaches['all_seats']['Coach']['CoachSeatPlan'],
$coaches['all_seats']['Coach']['CoachType'],
$coaches['all_seats']['Coach']['TicketFare'],
$coaches['all_seats']['Coach']['Decks']);
        $extra_data['seat_info'] = $seat_info;

        return $extra_data;
    }

    /**
     * Description: Function called by Lscrap to do the pre booking
     */
    public function preBookEBK($booking, $step) {

        $this->ci->lang->load('automatic_bookings');
        $this->ci->load->model('booking_events_tmp_model');

        try {
            /* Prepare data to send to API */
            if (! $booking_info = $this->prepareDataEBK($booking, $step)) throw
new BookingFail('prebookEBK no booking_info array');
            $booking_history['Whole booking'] = log2url($booking, 'booking');
            $booking_history['booking_info'] = log2url($booking_info, 'details');

            /* Call API to reserve the seat and obtain Booking Reference */
            $booking_reference = $this->easyBook("prebook", $booking_info, true);
            if(isset($booking_reference['error'])) {
                $booking_history['ebk_response'] =
log2url($booking_reference['error'], 'details');
                throw new BookingFail(' ERROR from EBK:
'.$booking_reference['error']['Message']);
```

```
            }
            if (! $booking_reference) throw new BookingFail('No seat reservation
(reserveseat) result from EBK API');

            /* Prepare $step */
            $step['bookingCode'] = $booking_reference['BookingReference'];
            $step['results'] = $booking_reference;

            /* Save the pre booking events on the DB */
            $this->ci->lang->set_language('english');
            $this->ci->booking_events_tmp_model->log_event($booking['code'],
'RoboEBK', 'message', $step['vessel'] . ": Ticket was SUCCESFULLY BOOKED With
PNR:" . $step['bookingCode'] . ' ' . log2url($booking_history, 'Booking
history'));
            $this->ci->lang->set_language($this->ci->user_language);
        } catch (BookingFail $e) {
            $this->ci->lang->set_language('english');
            $this->ci->booking_events_tmp_model->log_event($booking['code'],
'RoboEBK', 'message', $step['vessel'] . ": " . lang("automatic_booking_error") .
$e->getMessage() . ' ' . log2url($booking_history, 'Booking history'));
            $this->ci->lang->set_language($this->ci->user_language);
            $booking_history['FAILED'] = $e->getMessage();
            log_message('error', 'RoboEBK: (' . $booking['code'] . ') | ERROR: '
. $e->getMessage());
            return false;
        }

        return $step;
    }

    /**
     * Description: Function called by Ajax in order to block/release the seat
selected
     */
    public function toggleSeatKTM($step, $seat_id, $block) {
        return true;
    }

    /**
     * Description: Function called by Lscrap to do the booking
     */
    public function bookEBK($booking, $step) {

        $this->ci->lang->load('automatic_bookings');
        $this->ci->load->model('booking_events_model');
        $this->ci->load->model('bookings_model');
        $booking_history = array();

        try {

            if (! $reserveReference = $step['bookingCode'] ) throw new
BookingFail('No booking Code defined');
            $data = array("ReserveReference" => $reserveReference);
```

```
            /* Call the API and obtain results */
            // Book the seat in the API
            $booking_history['book_data_send_ebk']=log2url($data);
            $orderNumber = $this->easyBook("book",$data,true);
            if(isset($orderNumber['error'])) {
                $booking_history['book_data_error_ebk'] =
log2url($orderNumber['error'], 'details');
                throw new BookingFail(' ERROR from EBK:
'.$orderNumber['error']['Message']);
            }
            if(!$orderNumber) throw new BookingFail ('No booking result
(bookseat) from EBK API');
            $booking_history['book_result_ebk']=log2url($orderNumber);

            // Obtain Order Summary from API
            $booking_history['order_data_send_ebk']=log2url($data);
            if (! $results = $this->easyBook("order", $data, true) ) throw new
BookingFail('No order result (ordersummary) from EBK API');
            $booking_history['Issue results'] = log2url($results, 'details');

            /* Save the booking events on the DB */
            $this->ci->lang->set_language('english');
            $this->ci->booking_events_model->log_event($booking['code'],
'RoboEBK', 'message', $step['vessel'] . ": Ticket was SUCCESFULLY ISSUED With
PNR:" . $step['bookingCode'] . log2url($booking_history, 'Booking history'));
            $this->ci->lang->set_language($this->ci->user_language);

            /* Automatic booking and ticket generation */
            $url_ticket = $results['EticketUrl'];
            $this->generateVoucherEBK($url_ticket,$results,$booking,$step);

            $step['results'] = array('key' => 'value.... just to have something
into results and force step to update.');

            $this->ci->lang->set_language('english');
            $this->ci->booking_events_model->log_event($booking['code'],
'RoboEBK', 'message', $step['vessel'] . '<br><A HREF="' . $url_ticket . '">TICKET
TO PRINT</A>');
            $this->ci->lang->set_language($this->ci->user_language);
        } catch (BookingFail $e) {
            $this->ci->lang->set_language('english');
            $this->ci->booking_events_model->log_event($booking['code'],
'RoboEBK', 'message', $step['vessel'] . ": ISSUE FAILED" . $e->getMessage() . ' '
. log2url($booking_history, 'Booking history'));
            $this->ci->lang->set_language($this->ci->user_language);

            $booking_history['FAILED'] = $e->getMessage();
            log_message('error', 'RoboEBK: (' . $booking['code'] . ') | ISSUE
ERROR: ' . $e->getMessage());

            return false;
        }
```

```
        return $step;
    }
    /**
     * Description: Function that generates the voucher using a sample voucher
     * @param string $urlEBK
     */
    public function generateVoucherEBK($urlEBK,$results,$booking,$step) {

        /* Organize results to create the voucher */
        $grcodes=array();
        if (substr($urlEBK, 0, 5 ) != 'https') $urlEBK =
str_replace('http','https',$urlEBK);
        if( ! $htmlEBK = $this->getHTML($urlEBK,array("returnDOM"=>true) ) )
return false;
        foreach($htmlEBK->find('.qr-code') as $element) {
            $src = ($this->ci->is_production) ? "URL API PROD".$element->src :
"URL API TEST".$element->src ;
            $element->src = $src;
            $grcodes[] = $element->src;

        };

        /* Generate voucher per passenger */
        foreach ($results['TicketInfoList'] as $key => $ticketInfo) {

            //Organizing the data
            $itinerary=$ticketInfo['FromSubPlace']." -
".$ticketInfo['ToSubPlace'];
            $departure = DateTime::createFromFormat('Y-m-d
H:i:s',str_replace('T',' ',$ticketInfo['DepartureDate']))->format('d/m/Y H:i');
            $passengerName=($ticketInfo['PassengerType']=='ADULT')?
                $ticketInfo['PassengerName']." - "." Adult /
Dewasa":$ticketInfo['PassengerName']." - "." Child / Kanak-Kanak";

            $passport=$ticketInfo['PassengerIcOrPassport'];
            foreach ($step['seats'] as $seat)

if($seat['booking_params']['seat_number']==$ticketInfo['SeatNumber'])

$price=intval($seat['booking_params']['ticketPrice'][$ticketInfo['PassengerType']
]);

            $ids=$ticketInfo['ExternalTransactionID']." /
".$ticketInfo['TicketID'];
            $seatInfo=$ticketInfo['DeckCode']."/".$ticketInfo['SeatNumber'];

            $trainName= $step['extra_data']['booking_info']['train_name'];
            $coachName= $ticketInfo['Coach']." -
".$step['extra_data']['booking_info']['coach_name'][$ticketInfo['Coach']];

            // Generate the htmls for each passenger
```

```
            if(!$htmlVoucher =$this->getDOM(file_get_contents(APPPATH .
"libraries/Lscrap/ScrapLibEBK/voucherEBK.html")))return false;

            $htmlVoucher->getElementById('itinerary')->innertext =
strtoupper($itinerary);
            $htmlVoucher->getElementById('departure')->innertext = $departure;
            $htmlVoucher->getElementById('passengerName')->innertext =
$passengerName;
            $htmlVoucher->getElementById('passport')->innertext = $passport;
            $htmlVoucher->getElementById('ticketIds')->innertext = $ids;
            $htmlVoucher->getElementById('seatInfo')->innertext = $seatInfo;
            $htmlVoucher->getElementById('trainName')->innertext = $trainName;
            $htmlVoucher->getElementById('coachName')->innertext = $coachName;
            $htmlVoucher->getElementById('price')->innertext = "MYR
".intval(ceil(exchange($price,'SGD','MYR'))).".00 / L / Ordinary Ticket";

            foreach ($grcodes as $qrcode)
                if(strpos($qrcode, str_replace('-
','',$ticketInfo['TicketID']))!==false)
                    $htmlVoucher->getElementById('qr-code')->src=$qrcode;
            $htmls[]=$htmlVoucher;
        }

        // Get voucher path:
        $voucher_path = $this->ci->bookings_model->get_vouchers_path($booking);
        $voucher_file =  $voucher_path . '/' . $booking['code'] . '_' .
$step['bookingCode'] . '.pdf';

        // Load mPDF library
        $this->ci->load->library('m_pdf');
        $this->ci->m_pdf->initialize('"en-GB-x","A4","","",10,10,10,10,6,3');

        // Generate the PDF from the given html:
        for ($i=0;$i<count($htmls);$i++) {
            $this->ci->m_pdf->pdf->WriteHTML($htmls[$i]);
            $i!=(count($htmls)-1)?$this->ci->m_pdf->pdf->Addpage('P'):"";
        }

        // Save voucher:
        $this->ci->m_pdf->pdf->Output($voucher_file, 'F');
    }

    /**
     * Description: Gets the APIs token
     *
     * @param  boolean $force_relogin
     * @return array   $authentication the APIs token and signature
     */
    private function authenticationEBK($force_relogin = false) {

        $tokenfile = APPPATH . "tmp/ebkToken";
        $output=array();
```

```
        //TEMPORARY force prod for scrap only.
        $force_prod  = false;
        $creds = $this->ci->config->item('agent_accesses')['ebk'];
        $creds = ($this->ci->is_production || $force_prod)  ?  $creds['PROD'] :
$creds['TEST'] ;

        //Create signature
        $signature = md5("easybook" . $creds['SecureKey'] . $creds['Password']);
        $output['signature']=$signature;

        if (! file_exists($tokenfile) ) $force_relogin = true;
        else {
            $access_token = json_decode(file_get_contents($tokenfile), true);
            if ( strtotime($access_token['.expires']) < strtotime('-30 minutes')
) $force_relogin = true;
        }

        if ($force_relogin) {

            /* Obtain the token from the api */
            $urlToken = ($this->ci->is_production || $force_prod) ? "URL API
PROD": "URL API TEST";
            $html = $this->getHTML($urlToken, array(
                "postData"          => array(
                    "grant_type"    => "password",
                    "username"      => $creds['Username'],
                    "password"      => $creds['Password'] ))
                );

            /* Error handling */
            if (! $token = json_decode($html, true) ) return false;
            elseif ( isset($token['error']) ) {
                log_message('error', '[info] EBK could not log in: ' .
$token['error'] . ' description:' . $token['error_description']);
                return false;
            } elseif ( isset($token['access_token']) ) {
                file_put_contents($tokenfile, $html);
                $output['access_token']=$token['access_token'];
                return $output;
            }
        }
        else {
            $output['access_token']=$access_token['access_token'];
            return $output;
        }

    }
    /**
     * Description: function that calls the API to check for deposit
     * @return boolean
     */
    public function depositEBK() {
        if (! $balance = $this->easyBook("balance")) return false;
```

```
        return $balance['Account']['Balance'];
    }

    /**
     * Description: Function that does the calls to Easy Book's API
     * @param  string  $operation the type of operation we want to do in the API
     * @param  array   $data the parameters send to the API
     * @param  boolean $post if true, data is send via POST method
     * @return mixed if there are no errors while calling the API, it returns the
HTML as a JSON
     */
    private function easyBook($operation, $data = false, $post = false) {
        /* Models */
        $this->ci->load->model("logs_model");

        /* API Authentication */
        // Credentials
        $creds = $this->ci->config->item('agent_accesses')['ebk'];
        $creds = ($this->ci->is_production) ? $creds['PROD'] : $creds['TEST'];
        $url_base = $creds['endpoint'];

        if (! $authentication = $this->authenticationEBK()) return false;
        $signature = $authentication['signature'];


        $this->use_proxy = false;

        $ops2url = array(
            'companies'     => 'getcompanylist',
            'places'        => 'places',
            'nodes'         => 'subplaces',
            'routes'        => 'routes',
            'scrap'         => 'gettrips',
            'seats'         => $this->typeOfTransport == "train" ? 'querycoach' :
'queryseat',
            'fare'          => 'getbookingfare',
            'exactfare'     => 'getexactbookingfare',
            'prebook'       => 'reserveseat',
            'book'          => 'bookseat',
            'order'         => 'ordersummary',
            'balance'       => 'querybalance',
            'checkbooking'  => 'checkbooking',
        );

        $max_retries = 3;
        do {
            $max_retries -= 1;
            $access_token=$authentication['access_token'];

            /* Generate Authorization Header */
            $opt = array(
                    "headers"           => array("Authorization: Bearer " .
$access_token),
```

```
                    "getStatus"         => true,
                    "return_headers"    => true,
                    "timeout"           => 300
                );

            /* POST data */
            if ($post) $opt['postData'] = $data;
            if ($operation == 'order' || $operation == 'balance') {
                $opt['customRequest'] = "POST";
                array_push($opt['headers'], "Content-Length: 0");
                array_push($opt['headers'], "Accept: application/json");
            }

            $transport=($operation == 'balance' || $operation ==
'checkbooking')?"common":$this->typeOfTransport;

            /* Call the API */
            $url = $operation == 'order' ?
                $url_base . $transport . "/agent/" . $ops2url[$operation] .
"?sign=" . $signature . "&reserveReference=" . $data['ReserveReference'] :
                $url_base . $transport . "/agent/" . $ops2url[$operation] .
"?sign=" . $signature;

            // We dont call the API if we are in offline testing mode
            $offline_mode = APPPATH . 'tmp/testDataEBK/offline/' . $operation .
'_response_test';
            $response = !$this->ci->is_production && file_exists($offline_mode) ?
"testing mode..." : $this->getHtml($url, $opt);

            /* Save responses for test purposes in development */
            if (!$this->ci->is_production && file_exists(APPPATH .
'tmp/testDataEBK')) {
                $file = APPPATH . 'tmp/testDataEBK/' . date('Y-m-d') . '/' .
$operation;
                if (file_exists($offline_mode)) {
                    $response = json_decode(file_get_contents($offline_mode),
true);
                    $this->http_status = $response['http_status_code'];
                } else {
                    if (!file_exists($file)) mkdir($file, 0777, true);
                    $response['http_status_code'] = $this->http_status;
                    file_put_contents($file . '/url_' . date('H-i-s'), $url);
                    file_put_contents($file . '/request_' . date('H-i-s'),
json_encode($opt));
                    file_put_contents($file . '/response_' . date('H-i-s'),
json_encode($response));
                }
            }

            /* Error handling */
            $last_status = $this->http_status;
            try {
                if ($last_status == 200) {
```

```
                    if (count($response['headers']['content-type']) == 1 &&
preg_match('/\bxml\b/',
                            $response['headers']['content-type'][0]))
                        $html =
json_decode(json_encode(simplexml_load_string($response['body'],
'SimpleXMLElement',
                            LIBXML_NOCDATA)),
                            true);
                    elseif (count($response['headers']['content-type']) == 1 &&
preg_match('/\bjson\b/',
                            $response['headers']['content-type'][0]))
                        $html = json_decode($response['body'], true);
                    else throw new Exception(' Unsupported content-type in the
response', 2003);

                    /* API ERRORS */
                    if ($html['Status']) return $html;
                    else {
                        if (preg_match('@(SecurityToken|Security Token)@',
$html['Message'])) throw new Exception($html['Message'], 4001);
                        elseif (($html['Code'] == -5009 || preg_match('@(please
choose another seat)@',$html['Message']))) throw
                        new
Exception($html['Message'],intval("2002".abs($html['Code'])));
                        else throw new Exception($html['Message'],
intval("2001".abs($html['Code'])));
                    }
                } else {
                    // Access Denied or unauthorized error
                    if (preg_match('@access denied@i', $response['body']) ||
$last_status == 401)
                        throw new Exception(" Authentication error", 4001);
                    //Check if there is no data
                    if (!trim($response['body']))
                        throw new Exception(" No data in EBK response", 1001);
                    elseif (preg_match('@\berror\b@i', $response['body']))
                        throw new Exception(" Unknown error in the response while
calling API", 5001);
                    else {
                        sleep(rand(2, 5));
                        continue;
                    }
                }
            } catch (Exception $e) {
                $code = $e->getCode();
                $error_message = $e->getMessage();
                $error['Message'] = $error_message;

                /* API Error codes */
                if (substr($code, 0, 3) == 200) {
                    $code_temp = substr($code, 0, 4);
                    $code_api = str_replace($code_temp,'',$code);
                    $error['Code'] = $code_api;
```

```
                    if ($code_temp == 2001) $error['type'] = "API_ERROR";
                    elseif ($code_temp == 2002) {
                        //Seat Error
                        if (preg_match('/\bplease choose another seat\b/i',
$error_message) || intval($code_api) == 5009)
                            $error['type'] = "SEAT_NOT_AVAILABLE";
                        else $error['type'] = "SEAT_ERROR";

                    } elseif ($code_temp == 2003) $error['type'] = "
CONTENT_ERROR";
                    else $error['type'] = " UNKNOWN_API_ERROR";

                    log_message('error',
                        '[EBK] API Error => Type: ' . $error['type'] . ' Message:
' . $error['Message'] . " url: " . $url . ' - ' .
                        log2url($response['body'], 'html', 1));
                    $this->ci->logs_model->insert_log('API Message: ' .
$error['Message'], 'EBK', $operation . ' Failed');
                    $this->error = $error['Message'];
                    if ($operation == 'prebook' || $operation == 'book') return
array('error' => $error);

                    return false;
                } /* Codes that let us retry the request */
                elseif (substr($code, 0, 3) == 100) {
                    sleep(rand(3, 7));
                    log_message('error',
                        '[info] No data in EBK response, retrying... ' .
log2url($opt, 'request') . ' - ' . log2url($response,
                            'response') . ' - Last Status: ' . $last_status);
                    continue;
                } /* Authentication error codes */
                elseif (substr($code, 0, 3) == 400) {
                    if (!$this->authenticationEBK(true)) log_message('error',
                        '[EBK] Access denied. Failed to get new token ' . 'url: '
. $url . ' - ' . log2url($opt, 'request') .
                        ' - ' . log2url($response['body'], 'html') . ' - Last
Status: ' . $last_status);
                    else log_message('error',
                        '[info] Forced reauthentication in EBK succesfully: ' .
'url: ' . $url . ' - ' . log2url($response['body'],
                            'html'));
                    continue;
                } /* Unknown errors in the response */
                elseif (substr($code, 0, 3) == 500) {
                    log_message('error',
                        " [EBK] $error_message - url: $url" . ' - ' .
log2url($opt, 'request')
                        . ' - ' . log2url($response['body'], 'response', 1) . ' -
Last Status: ' . $last_status);
                    $this->ci->logs_model->insert_log('Message: ' .
$error['Message'], 'EBK', $operation . ' Failed');
```

```
                    $this->error = $error['Message'];
                    if ($operation == 'prebook' || $operation == 'book' ||
$operation == 'verifySeat' || $operation == 'checkBooking')
                        return array('error' => $error);

                    return false;
                } /* All other errors with no code defined we retry */
                else {
                    sleep(rand(2, 5));
                    log_message('error',
                        '[info] Unknown EBK response, retrying... ' .
log2url($opt, 'request') . ' - ' . log2url($response,
                            'response') . ' - Last Status: ' . $last_status);
                    continue;
                }
            }
        } while ($max_retries);

        $this->error = " API is not responding...";
        $this->ci->logs_model->insert_log('API may be down', 'EBK', $operation .
' Failed');
        log_message('error', "[EBK] API may be down... => url: $url - " .
log2url($opt, 'request') . ' - ' . log2url($response,
                'response') . ' - Last Status: ' . $last_status);

        return false;
    }turn false;
    }

    /**
     * Description: Function that builds towns, nodes and operational routes
using EBK API
     * @param array   $params the two towns
     * @param boolean $insert if set, (update or csv)
     */
    public function buildDataFromEBK($countryCode,$lastId, $operation, $params) {
        //Load Model and obtain data
        $this->ci->load->model("operational_routes_t_model");
        $this->ci->operational_routes_t_model->select('origin_id,origin_code')-
>as_array();
        $this->ci->operational_routes_t_model->distinct();
        $transportCode="'ebk'";
        $where = "transport_code =".$transportCode;
        $dataDB=$this->ci->operational_routes_t_model->get_many_by($where);

        $idDB = $lastId; // Last ID on the DB

        if( ! $subplacesEBK = $this->getSubplacesEBK($countryCode,$params) )
return false;

        if (!$params) {
            /* Build the towns and nodes */
            $output=$this->buildTownsNodes($subplacesEBK['nodes'], $lastId);
```

```
            $towns=$output['towns'];
            $nodes=array_values($output['nodes']);

            /* Build the operational routes */
            $operationalRoutes=$this-
>buildOperationalRoutes($output['nodes'],$subplacesEBK['nodes'],$subplacesEBK['ro
utesEBK']);
        }

        /* Insert new data */
        if ($operation) {

            if ($operation == 'new' && $params) {
                $newData=explode(':',urldecode($params));

                /* Build the towns and nodes */
                if( ! $output = $this->buildTownsNodes($subplacesEBK['nodes'],
$idDB,$newData) ) return false;

                $buildData['towns']=$output['towns'];
                $buildData['nodes']=array_values($output['nodes']);

                /* Build the operational routes */
                $buildData['operational_routes'] = $this-
>buildOperationalRoutes($output['nodes'],
$subplacesEBK['nodes'],$subplacesEBK['routesEBK'],$dataDB);
                $this->editBuildData($buildData);
            }

            //Insert to DB
            if ($operation == "update") {
                $buildData['towns'] = $towns;
                $buildData['nodes'] = $nodes;
                $buildData['operational_routes'] = $operationalRoutes;
                $this->editBuildData($buildData);
            }

            //Save as CSV file
            else if ($operation == "csv") {
                $file = APPPATH . "libraries/Lscrap/ScrapLibEBK/";

            // Towns
            $townsFile = $file . "towns.csv";
            $fpTown = fopen($townsFile, 'w');
            fputcsv($fpTown, array_keys(array_values($towns)[0]));
            foreach ($towns as $town)
                fputcsv($fpTown, $town);
            fclose($fpTown);

            // Nodes
            $nodesFile = $file . "nodes.csv";
            $fpNodes = fopen($nodesFile, 'w');
```

```
            fputcsv($fpNodes, array_keys(array_values($nodes)[0]));
            foreach ($nodes as $node)
                fputcsv($fpNodes, $node);
            fclose($fpNodes);

            // Operational Routes
            $operationalRoutesFile = $file . "operationalRoutes.csv";
            $fpRoutes = fopen($operationalRoutesFile, 'w');
            fputcsv($fpRoutes, array_keys(array_values($operationalRoutes)[0]));
            foreach ($operationalRoutes as $route)
                fputcsv($fpRoutes, $route);
            fclose($fpRoutes);

            echo "OK: data inserted to CSV files, look in: " . $file;
            }
        }

        // Show the data
        else {
            echo "Towns created: ".count($towns)." ".log2url($towns,'Towns') .
"</br>";
            echo "Nodes created: ".count($nodes)." " . log2url($nodes,'Nodes') .
"</br>";
            echo "Operational routes created: ".count($operationalRoutes)."
".log2url($operationalRoutes,'Operational Routes') . "</br>";
            file_put_contents(APPPATH . 'tmp/towns.json',json_encode($towns));
            file_put_contents(APPPATH . 'tmp/nodes.json',json_encode($nodes));
            file_put_contents(APPPATH .
'tmp/op.json',json_encode($operationalRoutes));
        }
    }
    private function
buildOperationalRoutes($nodes,$nodesEBK,$routesEBK,$newData=false) {
        if(!$newData) {
            foreach ($routesEBK as $route){
                if( isset($nodes[$route['FromSubPlaceId']]) &&
isset($nodes[$route['ToSubPlaceId']]) && $route['FromSubPlaceId'] !=
$route['ToSubPlaceId'] ) {
                    $operationalRoutes[] = array(
                        'origin_id'         =>
$nodes[$route['FromSubPlaceId']]['id'],
                        'destination_id'    =>
$nodes[$route['ToSubPlaceId']]['id'],
                        'origin_code'       =>
$route['FromSubPlaceId'].":".$route['FromPlaceId'],
                        'destination_code'  =>
$route['ToSubPlaceId'].":".$route['ToPlaceId'],
                        'transport_code'    => 'ebk',
                        'deleted'           => 0
                    );
                }
            }
        }
```

```
        else {
            foreach ($routesEBK as $route){
                if( in_array( $route['FromSubPlaceId'], array_keys($nodes) ) ||
in_array( $route['ToSubPlaceId'], array_keys($nodes) ) ) {

if(array_search($route['FromSubPlaceId'],array_column($nodesEBK, 'town_id')) &&

array_search($route['ToSubPlaceId'],array_column($nodesEBK, 'town_id'))) {

                        //Origin

$origin_code=$route['FromSubPlaceId'].":".$route['FromPlaceId'];

$origin_key=array_search($origin_code,array_column($newData, 'origin_code'));
                        if(in_array($route['FromSubPlaceId'],array_keys($nodes)))
                            $origin_id=$nodes[$route['FromSubPlaceId']]['id'];
                        else

$origin_id=$origin_key?$newData[$origin_key]['origin_id']:false;

                        //Destination

$destination_code=$route['ToSubPlaceId'].":".$route['ToPlaceId'];

$destination_key=array_search($destination_code,array_column($newData,
'origin_code'));
                        if(in_array($route['ToSubPlaceId'],array_keys($nodes)))
                            $destination_id=$nodes[$route['ToSubPlaceId']]['id'];
                        else

$destination_id=$destination_key?$newData[$destination_key]['origin_id']:false;

                        if($origin_id && $destination_id && $origin_code !=
$destination_code) {
                            $operationalRoutes[] = array(
                                'origin_id'         => intval($origin_id),
                                'destination_id'    => intval($destination_id),
                                'origin_code'       => $origin_code,
                                'destination_code'  => $destination_code,
                                'transport_code'    => 'ebk',
                                'deleted'           => 0
                            );
                        }
                    }
                }
            }
        }

        $operationalRoutes =
isset($operationalRoutes)?array_values(array_unique($operationalRoutes,
SORT_REGULAR)):array();

        return $operationalRoutes;
```

```
    }

    public function get_valid_town_code ($town, $newCodes = array()) {
       $this->ci->load->model("Towns_model");
       $this->ci->Towns_model->select('code')->as_array();
       $codesDB=$this->ci->Towns_model->get_all();

       foreach ($codesDB as $codeDB) $codes[]=array_values($codeDB)[0];
       $iatas =
json_decode(file_get_contents(APPPATH.'tmp/asianiatas.txt'),true);
       $cont = 5000;

       do {
              $cont -= 1;
              srand($cont);
              $shuffle=substr(str_shuffle(str_replace(' ', '',strtoupper($town))),
0, 3);
              if (in_array($shuffle, $codes)) continue;
              if (in_array($shuffle, $iatas)) continue;
              if (in_array($shuffle, $newCodes)) continue;
              return $shuffle;


       } while ( $cont ) ;
    }

    private function buildTownsNodes($nodesEBK,$idDB,$newData=false) {
        //Load Model and obtain data
        $this->ci->load->model("Towns_model");

        $output=array();
        $newcodes = array();
        $cont=1;

        foreach ($nodesEBK as $nodeEBK) {
             $id = intval($idDB + $cont);
            $code =  $this->get_valid_town_code($nodeEBK['town_en'], $newcodes);
            $newcodes[] = $code;
            $rem = false;

            //check if the town already exist;
            if ($dbTown = $this->ci->Towns_model->get_by(array('name_en' =>
strtoupper($nodeEBK['town_en']) ))) $rem = true;

            if (!$newData) {

                $output['nodes'][$nodeEBK['town_id']] = array(
                    "id"           => $id,
                    "node_type"    => "train",
                    "name_en"      => $nodeEBK['town_en']." Railway Station",
                    "name_vi"      => "Ga ".$nodeEBK['town_en'],
                    "name_ja"      => $nodeEBK['town_en']." 鉄道駅",
```

```
                "name_cn"      => $nodeEBK['town_en']." 火车站",
                "name_tw"      => $nodeEBK['town_en']." 火車站",
                "town_en"      => strtoupper($nodeEBK['town_en']),
                "town_vi"      => strtoupper($nodeEBK['town_en']),
                "town_ja"      => strtoupper($nodeEBK['town_en']),
                "town_cn"      => strtoupper($nodeEBK['town_en']),
                "town_tw"      => strtoupper($nodeEBK['town_en']),
                "country_code" => $nodeEBK['country_code'],
                "latitude"     => $nodeEBK['latitude'],
                "longitude"    => $nodeEBK['longitude'],
                'deleted'      => 0
            );
            if ($rem) {
                $cont++;
                continue;
            }

            $output['towns'][] = array(
                "name_en"      => strtoupper($nodeEBK['town_en']),
                "name_vi"      => strtoupper($nodeEBK['town_en']),
                "name_ja"      => strtoupper($nodeEBK['town_en']),
                "name_cn"      => strtoupper($nodeEBK['town_en']),
                "name_tw"      => strtoupper($nodeEBK['town_en']),
                'pivots'       => 'KUALA LUMPUR',
                'country_code' => $nodeEBK['country_code'],
                'town_type'    => '1',
                'code'         => $code,
                'deleted'      => 0
            );

            $cont++;
        }
        else if( $newData && in_array($nodeEBK['town_en'], $newData) ) {

         $output['nodes'][$nodeEBK['town_id']] = array(
                "id"           => $id,
                "node_type"    => "train",
                "name_en"      => $nodeEBK['town_en']." Railway Station",
                "name_vi"      => "Ga ".$nodeEBK['town_en'],
                "name_ja"      => $nodeEBK['town_en']." 鉄道駅",
                "name_cn"      => $nodeEBK['town_en']." 火车站",
                "name_tw"      => $nodeEBK['town_en']." 火車站",
                "town_en"      => strtoupper($nodeEBK['town_en']),
                "town_vi"      => strtoupper($nodeEBK['town_en']),
                "town_ja"      => strtoupper($nodeEBK['town_en']),
                "town_cn"      => strtoupper($nodeEBK['town_en']),
                "town_tw"      => strtoupper($nodeEBK['town_en']),
                "country_code" => $nodeEBK['country_code'],
                "latitude"     => $nodeEBK['latitude'],
                "longitude"    => $nodeEBK['longitude'],
                'deleted'      => 0
```

```
            );

            if ($rem) {
                $cont++;
                continue;
            }

               $output['towns'][] = array(
                   "name_en"      => strtoupper($nodeEBK['town_en']),
                   "name_vi"      => strtoupper($nodeEBK['town_en']),
                   "name_ja"      => strtoupper($nodeEBK['town_en']),
                   "name_cn"      => strtoupper($nodeEBK['town_en']),
                   "name_tw"      => strtoupper($nodeEBK['town_en']),
                   'pivots'       => 'KUALA LUMPUR',
                   'country_code' => $nodeEBK['country_code'],
                   'town_type'    => '1',
                   'code'         => $code,
                   'deleted'      => 0
               );

               $cont++;
           }
       }
       if ( ! isset($output['towns']) ) $output['towns']  = array ();
       if ( ! isset($output['nodes']) ) $output['nodes']  = array ();
       return $output;
   }

   private function editBuildData($buildData) {
       /* Load the models */
       $this->ci->load->model("Towns_model");
       $this->ci->load->model("Nodes_model");
       $this->ci->load->model("operational_routes_t_model");

       // Updatable fields:
       $fields = array('towns','nodes','operational_routes');

       // Run through the fields to collect data:
       $updated_data = array();
       foreach ($fields as $field) {
           // Check if we are saving the data:
           if ($this->ci->input->post($field)) {
               // Decode JSON data:
               $json = preg_replace("#[\r\n]#", '', $this->ci->input-
>post($field));
               $data = json_decode($json, true);
               if (! $data)
                   echo 'Invalid JSON';
               else
                   $updated_data[$field] = $buildData[$field] = $data;
           }
       }
```

```
        // Open the form:
        echo '<div style="width:70%; margin:50px auto">';
        echo "<h1>Edit Data to Insert in DB</h1>";
        echo form_open();

        // Fields:
        foreach ($fields as $field) {
            echo
form_label("<strong>$field:".count($buildData[$field])."</strong>", $field) .
"<br>";
            echo form_textarea(array(
                'name' => $field,
                'value' => json_encode($buildData[$field], JSON_PRETTY_PRINT),
                'style' => 'width:100%; height:200px; margin:auto'
            ));
            echo "<br><br>";
        }

        // Close the form:
        echo form_submit('submit', 'Update and Insert');
        echo form_close();

        $towns = $buildData['towns'];
        $nodes = $buildData['nodes'];
        $operationalRoutes = $buildData['operational_routes'];

        if($updated_data)
        {
            /* Insert data to DB */
            $this->ci->Towns_model->insert_batch($updated_data['towns']);
            $sqlTowns=$this->ci->db->last_query();

            $this->ci->Nodes_model->insert_batch($updated_data['nodes']);
            $sqlNodes=$this->ci->db->last_query();

            $this->ci->operational_routes_t_model-
>insert_batch($updated_data['operational_routes']);
            $sqlOR=$this->ci->db->last_query();

            file_put_contents(APPPATH . "libraries/Lscrap/ScrapLibEBK/sql.txt",
"Towns: ".PHP_EOL." $sqlTowns",FILE_APPEND);
            file_put_contents(APPPATH . "libraries/Lscrap/ScrapLibEBK/sql.txt",
"Nodes: ".PHP_EOL." $sqlNodes",FILE_APPEND);
            file_put_contents(APPPATH . "libraries/Lscrap/ScrapLibEBK/sql.txt",
"Operational routes: ".PHP_EOL." $sqlOR",FILE_APPEND);

            echo "Actualizado e insertado en DB";
        }
        // Close the container:
        echo "</div>";
    }
}
```

5.3.3 CÓDIGO TAILANDIA

```
<?php defined('BASEPATH') OR exit('No direct script access allowed');

trait ScrapLibRRT {

    private $train2vessel = array(
        'Rapid' => 'RP',
        'Express' => 'EX',
        'SpecialExpress' => 'SP',
        'Tour' => 'TR',
        'Ordinary'  => 'OR',
        ""=>'',
    );
    private $service2fare = array(
        'Air-conditioned Power Diesel' => 'temp',
        'Air-Conditioned First Class' => '1AnL',
        'Air-Conditioned Second Class' => '2BnL',
        'Bogie Second Class' => '2Bn',
        'Bogie Third Class' => '3',
        '2Seat' => '2NM',
        '3Seat' => '3NC',
        'Room' => 'SGR',
        'Upper Bed' => 'T2',
        'Lower Bed' => 'T1',
    );

    function scrapRRT($params)
    {

       //Constants and config
       $this->ci->load->model('exchange_rates_model');
     //  $exchange_rate = $this->ci->exchange_rates_model-
>exchange_rates['THB/VND'];
       $origin = $params['origin'];
       $destination = $params['destination'];
       $departure_date = $params['departure_date'];

       $url="URL API";

       //First retrieve the schedules and vessel name
       if (!$html =$this->getHTML($url,array("postData" => array("StationFirst"
=> ($origin),
           "StationLast" => $destination,
           "Submit"=>"Check"),"returnDOM"=>true))) {
                   log_message('error', '[info] RRT scrap returned no html for
this '.log2url($params, 'route'));
                   return false;
       }
       $rows = array();
       $cols = array();
       $trains = array();
```

```
       $routes = array();

      // print_array ($routes);

       // Obtain all the rows of the table
       foreach($html->find('table') as $element){
           if($element->title=="This is a table of train number , timetable and
check fare.")
           {
               foreach ($element->find('tr') as $item)
               {
                   array_push($rows,$item->plaintext);
                   foreach ($item->find('td') as $result)
                       array_push($cols,$result->plaintext);
               }
           }
        };
        for($i=0;$i<count($cols);$i++)
        {
            //Column obtained in the DOM
            $arrival= 7*$i+4;
            $departure=7*$i+3;
            $typeTrain=7*$i+2;
            $trainNum=7*$i+1;

            if($trainNum<count($cols) && $departure<count($cols) &&
$arrival<count($cols))
            {
                $duration= (strtotime(trim($cols{$arrival}))-
strtotime(trim($cols{$departure})) > 0 ) ?
                (strtotime(trim($cols{$arrival}))-
strtotime(trim($cols{$departure})))/60 :
                (strtotime(trim($cols{$arrival}).' +1 day')-
strtotime(trim($cols{$departure})) )/60;

                $type = str_replace(" ","",trim($cols{$typeTrain}));
                $vessel = trim($this->train2vessel[$type]).str_replace("
","",trim($cols{$trainNum},"</font>"));

                array_push($trains,array(
                   'vessel' => $vessel,
                   'start_time' =>
date('H:i:s',strtotime(trim($cols{$departure}))),
                   'duration' => $duration));
            }
        };

        //Trains obtained without duplicates
        $trains = array_unique($trains,SORT_REGULAR);

        $countTrain=0;
```

```
        //Obtain url to check the price for each train
        foreach($html->find('a') as $element){

            if(stristr($element->href, "URL API"))
            {
                $urlPrice = $element->href;
                $urlPrice = str_replace(" ","%20", $urlPrice);

               if (!$htmlPrice = $this->getHTML($urlPrice,array("returnDOM" =>
true))){
                        log_message('error', '[info] RRT scrap Could not get
prices for vessel '.$trains[$countTrain]['vessel']);
                            continue;
               }

                $fares = array();

                //Fare types
                foreach ($htmlPrice->find('.style127') as $item)
                    array_push($fares,trim($item->plaintext));

                //Prices
                $prices=array();
                foreach ($htmlPrice->find('td') as $result)
                {
                   preg_match('/Adult (.*?) Child/', stristr(trim($result-
>plaintext), "Adult"), $match);
                    if(count($match)>1)
                    {
                        $data=str_replace(',', '', $match[count($match)-1]);
                        array_push($prices,array_values(array_filter(explode(" ",
                                    preg_replace("/[^0-9]/","
",trim($data))))));
                    }
                }
               $prices=array_values(array_unique($prices,SORT_REGULAR));


               //Ordering the prices with each fare
               $data=array();
               for($i=0;$i<count($fares);$i++)
                {
                    unset($codigo);

                    if(stristr($fares[$i],"Air-conditioned Power Diesel"))
                        $codigo=$this->service2fare['Air-conditioned Power
Diesel'];

                    if(stristr($fares[$i],"Air-Conditioned First Class"))
                        $codigo=$this->service2fare['Air-Conditioned First
Class'];
```

```
                    if(stristr($fares[$i],"Air-Conditioned Second Class"))
                        $codigo=$this->service2fare['Air-Conditioned Second
Class'];

                    if(stristr($fares[$i],"Bogie Second Class"))
                        $codigo=$this->service2fare['Bogie Second Class'];

                    if(stristr($fares[$i],"Bogie Third Class"))
                        $codigo=$this->service2fare['Bogie Third Class'];

                    if(stristr($fares[$i],"Bogie Power Diesel Railcar - THN."))
                          $codigo=$this->service2fare['Bogie Third Class'];

                    if ( !isset( $codigo ) ) $codigo=$this->service2fare['Bogie
Third Class'];

                        //Sit
                        if(count($prices[$i])==1)
                                        {
                            if($codigo=="temp")
                                array_push($data,$prices[$i][0]);

                            if($codigo=="2Bn")
                                array_push($data,$this-
>service2fare['2Seat'].':'.intval($prices[$i][0]));
                            elseif($codigo=="3")
                                array_push($data,$this-
>service2fare['3Seat'].':'.intval($prices[$i][0]));
                        }
                        //Upper Lower
                        if(count($prices[$i])==2)
                        {
                            array_push($data,$codigo.$this->service2fare['Upper
Bed'].':'.intval($prices[$i][0]));
                            array_push($data,$codigo.$this->service2fare['Lower
Bed'].':'.intval($prices[$i][1]));
                        }
                        //Upper Lower and Room
                        if(count($prices[$i])==3)
                        {
                            array_push($data,$codigo.$this->service2fare['Upper
Bed'].':'.intval($prices[$i][0]));
                            array_push($data,$codigo.$this->service2fare['Lower
Bed'].':'.intval($prices[$i][1]));
                            array_push($data,$this-
>service2fare['Room'].':'.intval($prices[$i][2]));
                        }
                 }
                $extra_data = array('check_availability' => array());
                $prices_status = array();


                foreach ($data as  $d) {
```

```
                    $k = explode(':', $d)[0];
                    $prices_status[$k]=1;
                          $extra_data['check_availability'][$k] = 9;
                }
                $preciosFinal=implode('#',$data);

                //Lowest price
                $price="";
                asort($prices);
                if(count($prices)>0)
                {
//                  $price = filter_var(array_values($prices)[0],
FILTER_SANITIZE_NUMBER_INT);
                    $price=min(min($prices));
                }
                //Obtaining the routes $routes =
array_merge($trenes,$aniadoprecioatrenes);
                array_push($routes,array_merge($trains[$countTrain],array(
                    "price" =>  intval($price),
                    "prices" => $preciosFinal,
                       "currency"  => 'THB',
                       "prices_status" => $prices_status,
                       "extra_data" => $extra_data,
                       'transport_code' => 'rrt',
                ));
                $countTrain++;
            }
        };

        if ( strtotime($departure_date) < strtotime('2018-09-30') &&
in_array(date("D", strtotime($departure_date)), array('Sat', 'Sun' ))){
             if ($origin == 'Bangkok' && in_array($destination, array('Ban Plu Ta
Luang','Pattaya'))) {
                    foreach ($routes as $k => $v) if ($v['vessel'] == 'RP997')
unset($routes[$k]);
                    $routes[] = array(
                                'vessel' => 'RP997',
                                'start_time' => '06:45:00',
                                'duration' => ($destination == 'Ban Plu Ta
Luang' ) ? 185 : 148 ,
                                'price' => 170,
                                'prices' => '2NML:170',
                                'currency' => 'THB',
                                'transport_code' => 'rrt',
                                'extra_data' => array('check_availability' =>
array('2NML' => 9)),
                                'prices_status' =>  array ('2NML' => 1),
                    );
             } else if ($destination == 'Bangkok' && in_array($origin, array('Ban
Plu Ta Luang','Pattaya'))) {
                    foreach ($routes as $k => $v) if ($v['vessel'] == 'RP998')
unset($routes[$k]);
                    $routes[] = array(
```

```
                                    'vessel' => 'RP998',
                                    'start_time' => ($origin == 'Ban Plu Ta Luang' )
? '15:50:00' : '16:26:00',
                                    'duration' => ($origin == 'Ban Plu Ta Luang' ) ?
185 : 149 ,
                                    'price' => 170,
                                    'prices' => '2NML:170',
                                    'currency' => 'THB',
                                    'transport_code' => 'rrt',
                                    'extra_data' => array('check_availability' =>
array('2NML' => 9)),
                                    'prices_status' =>  array ('2NML' => 1),
                        );
                }

        }

        //echo json_encode($routes);
        return $routes;
    }
}
```

Capítulo 6. ANALISIS DE RESULTADOS

6.1 RESULTADOS EXTRACCION AUTOMATICA DE DATOS

En las siguientes figuras se muestran los resultados, en formato JSON, de la extracción automatica de información sobre los nodos, asi como la creación automatica de rutas.

6.1.1 EXTRACCIÓN DE INFORMACION DE LOS NODOS

Por ejemplo, si se esta buscando la estación de tren de Ipoh, se llama a las funciones descritas en 5.2.1 y los resultados que se obtienen son los siguientes:


```
{
  "latitude": "4.59906255",
  "longitude": "101.073891452714",
  "urlSearch": "https://nominatim.openstreetmap.org/search?q=Ipoh&countrycodes=MY&accept-language=en-US&format=json&addressdetails=1",
  "message": "OK"
}
```


Figura 16. Resultados extraccion de coordenadas


```
{
  "English": "Ipoh",
  "Japanese": "イポー",
  "Chinese": "怡保",
  "Simplified Chinese": "怡保",
  "Traditional Chinese": "怡保"
}
```


Figura 17. Resultados extraccion de idiomas

Al comprobar las coordenadas con Google Maps, se observa que los resultados son los deseados.

6.1.2 CREACION AUTOMATICA DE RUTAS

Para la creación automatica de las rutas, se llama a la función buildDataFromEBK descrita en 5.2.2 y los resultados que se obtienen son los siguientes:


```
{
  "id": 1515,
  "node_type": "train",
  "name_en": "Ipoh Railway Station",
  "name_vi": "Ga Ipoh",
  "name_ja": "イポー駅",
  "name_cn": "怡保站",
  "name_tw": "怡保站",
  "town_en": "IPOH",
  "town_vi": "IPOH",
  "town_ja": "イポー",
  "town_cn": "怡保",
  "town_tw": "怡保",
  "country_code": "MY",
  "latitude": "4.59906255",
  "longitude": "101.073891452714",
  "deleted": 0
},
```


Figura 18. Resultados creacion de nodo


```
{
  "origin_id": 1518,
  "destination_id": 1519,
  "origin_code": "180:2",
  "destination_code": "181:2",
  "transport_code": "ebk",
  "deleted": 0
},
{
  "origin_id": 1518,
  "destination_id": 1526,
  "origin_code": "180:2",
  "destination_code": "300:2",
  "transport_code": "ebk",
  "deleted": 0
},
```


Figura 19. Resultados creacion de ruta operacional

6.2 RESULTADOS DE MALASIA

En este apartado, se reflejan los resultados de la integración del proceso de reserva al buscador.

6.2.1 BUSQUEDA DE RUTAS

Aquí, el buscador llama a la función scrap descrita en la sección 5.2.2, donde se obtienen los resultados entre el origen y el destino.

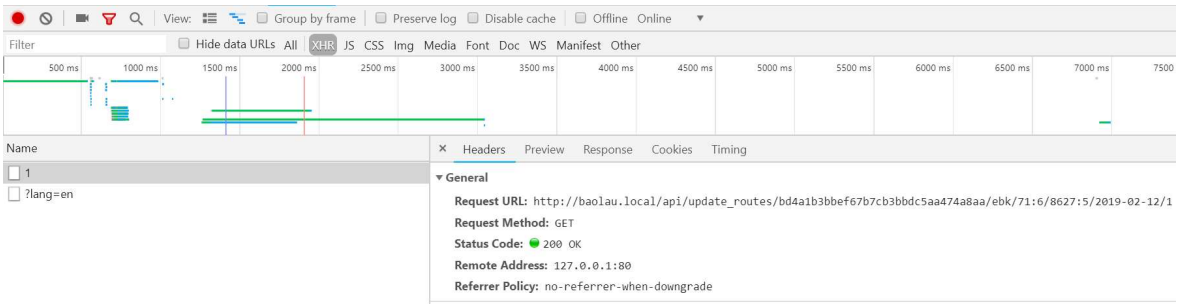


Figura 20. Llamada a la funcion scrap

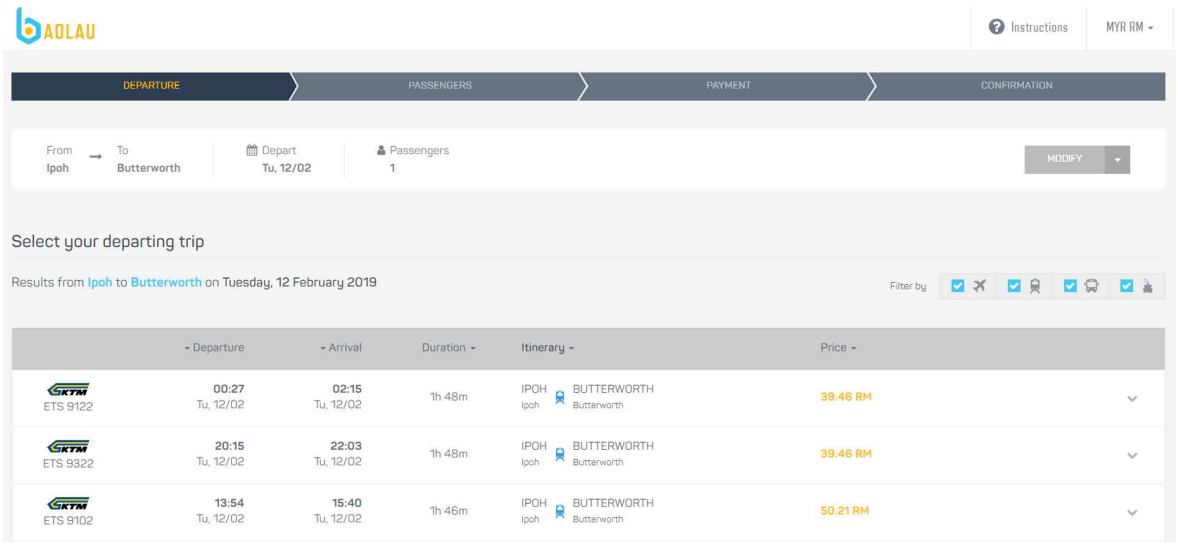


Figura 21. Resultados de la funcion scrap

6.2.2 CREACION DEL MAPA DE ASIENTOS

Una vez obtenidos los resultados, se busca la información necesaria de los asientos: disponibilidad, mapa de asientos, etc. Esta información se obtiene llamando a la función trainSeatMap descrita en 5.2.2 y los resultados se guardan en el formato de Baolau.

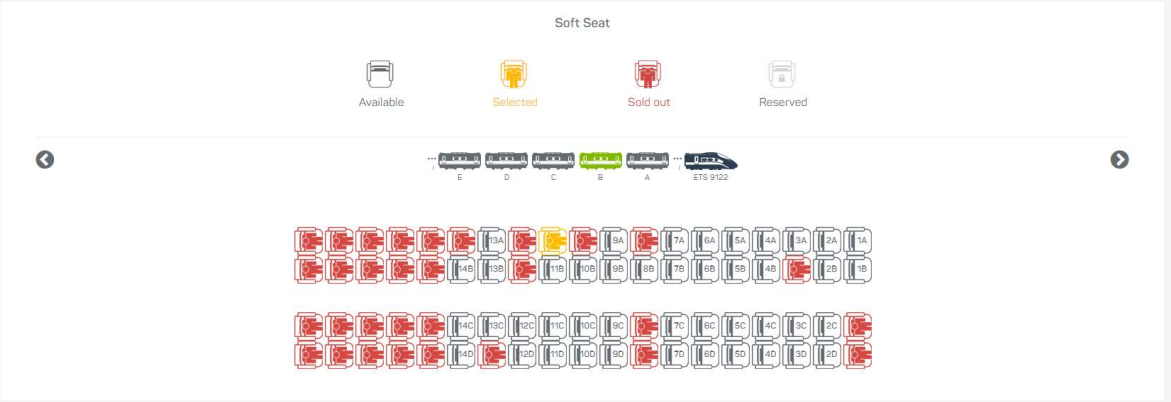


Figura 22. Resultados mapa de asientos

6.2.3 RESERVA DE ASIENTO

Una vez finalizado el proceso anterior, se hace la reserva del asiento llamando a la función preBook. Esta función se encarga de organizar los datos introducidos por el cliente y formatearlos de tal forma que se puedan enviar al proveedor para asi proceder con la reserva. Cuando temina el preBook, se llama a la función book que se encarga de finalizar la reserva y generar los billetes.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

Gracias a este Trabajo Fin de Máster he conocido y usado varias herramientas que facilitan el desarrollo software de una forma eficiente y eficaz. Cabe destacar que con el proyecto también he aprendido a utilizar herramientas que facilitan la automatización de extracción de datos.

En este apartado tratare sobre las conclusiones del proyecto, el aprendizaje obtenido, los objetivos cumplidos, los logros y dificultades encontrados y las posibles mejoras.

7.1 APRENDIZAJE

El mayor aprendizaje que he obtenido al realizar el proyecto es el de conocer herramientas que ayudan a automatizar la extracción de datos. También he aprendido nuevos métodos y herramientas que son útiles a la hora de codificar, hacer pruebas de integracion, para gestionar y construir proyectos y para hacer controles de versiones del código.

7.2 OBJETIVOS CUMPLIDOS

Atendiendo a los objetivos marcados inicialmente al comienzo del proyecto, veo que he conseguido cumplirlos. Con este proyecto, se ha conseguido, de una manera cómoda y eficiente, automatizar la extracción de información sobre las distintas estaciones a integrar en el buscador, asi como automatizar el proceso de reserva y compra de billetes de todas las estaciones integradas.

7.3 LOGROS Y DIFICULTADES ENCONTRADAS

Los mayores logros que se han conseguido con este proyecto son:

- Minimizar el coste de desarrollo: esto quiere decir que los errores que pueden surgir sean detectados y solucionados a tiempo
- Maximizar la calidad: evitar que las versiones del código nuevo no estropeen al código que ya estaba implementado.

Una de las mayores dificultades encontradas durante el desarrollo de este proyecto fue a la hora de automatizar la información necesaria en la busquedad de los nodos ya que muchos de ellos se encontraban en zonas remotas y no poseian traducción en ninguno de los idiomas requeridos.

7.4 POSIBLES MEJORAS

Una posible mejora que implementar en el proyecto es añadir el testeo al sistema con herramientas que automaticen la interacción del usuario.

Capítulo 8. BIBLIOGRAFÍA

[1] Media Wiki API help. https://www.wikidata.org/w/api.php.

[2] Web Scraping. https://es.wikipedia.org/wiki/Web_scraping.

[3] CodeIgniter Active Record. https://codeigniter.com/userguide2/database/active_record.html

[4] cURL. https://curl.haxx.se

[5] Railway travel blog. https://railtravelstation.com/

[6] HTML to DOM. http://simplehtmldom.sourceforge.net/manual.htm

[7] Learning PHP. https://laracasts.com/series/php-for-beginners